

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Шифр «Автоматизація тестування»

СИСТЕМА АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ ЗНАНЬ

2019 – 2020 н.р.

ЗМІСТ

ВСТУП	3
РОЗДІЛ 1. СТРУКТУРА АВТОМАТИЗОВАНОЇ СИСТЕМИ ТЕСТУВАННЯ ЗНАНЬ	4
РОЗДІЛ 2. ІНТЕРФЕЙС ТА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТЕСТУВАННЯ ЗНАНЬ	18
ВИСНОВКИ	26
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	27
Додаток А.....	28

ВСТУП

Нині інформаційні технології стали важливою частиною людського суспільства. Існування сучасної освітньої системи необхідно для розвитку інформаційного суспільства. Тому використання інформаційних технологій в освітній системі допомагає вирішити одразу кілька важливих завдань:

- розвиток інформаційних технологій в суспільстві;
- розвиток освіти за допомогою сучасних інформаційних технологій.

Сфера освіти, як і будь-яка інша, постійно розвивається. Разом з нею з'являються нові форми навчання та технології. Одним із головних напрямів удосконалення навчання є розробка автоматизованої системи контролю знань, що дозволяє ефективно та швидко оцінювати знання. Комп'ютерне тестування є сучасним, адаптивним засобом для перевірки успішності студентів. Розвиток інформаційних технологій дозволив впровадити сучасні комп'ютерні технології в процес контролю і оцінювання успішності. Отже головним завданням роботи є дослідження процесу тестування знань та створення сучасної програми для автоматизації цього процесу. Незважаючи на існуючі розробки у цієї галузі дане завдання є актуальним. Адже з одного боку змінюються вимоги щодо організації начального процесу, зокрема вимоги щодо контролю знань, з іншого – комп'ютерні технології також набувають змін. Система тестування має бути універсальною та забезпечувати середовище як для розробників тестів так і для тих, хто тестується. Користувач повинен мати змогу створювати та редагувати завдання тестів, складати тести з банку завдань тощо. Також система має забезпечувати певний захист даних и таке інше.

Для реалізації завдання обрано мову програмування C# [4-6] та середовище розробки Visual Studio 2017. Для реалізації системи було вирішено використовувати паттерн Singleton [1]. Інтерфейс користувача проектувався засобами Windows Forms [2, 3], що має велику кількість компонентів для візуального представлення функцій програми та їх ефективним управлінням.

РОЗДІЛ 1. СТРУКТУРА АВТОМАТИЗОВАНОЇ СИСТЕМИ ТЕСТУВАННЯ ЗНАНЬ

Комп'ютерну систему тестування знань реалізовано як десктоп-застосунок. Програмний засіб дає можливість створення бази тестів, проведення тестування та розрахунок і експорт результатів тестування. До головних модулів програми належать: модуль створення та редагування бази тестів, модуль проходження тестування, модуль збереження та зчитування бази тестів та модуль захисту бази тестів. Система має забезпечити зручне складання, зберігання тестів в пам'яті комп'ютера та проведення тестування. Після проведення тестування результати мають виводитися у вигляді документа на друк. Також нами передбачено систему захисту інформації, яка не дозволить переглядати та редагувати тести користувачами без прав доступу. На рис. 1.1-1.2 наведено діаграму прецедентів для створеної нами комп'ютерної системи тестування.



Рисунок 1.1. Варіанти використання ПЗ для студента

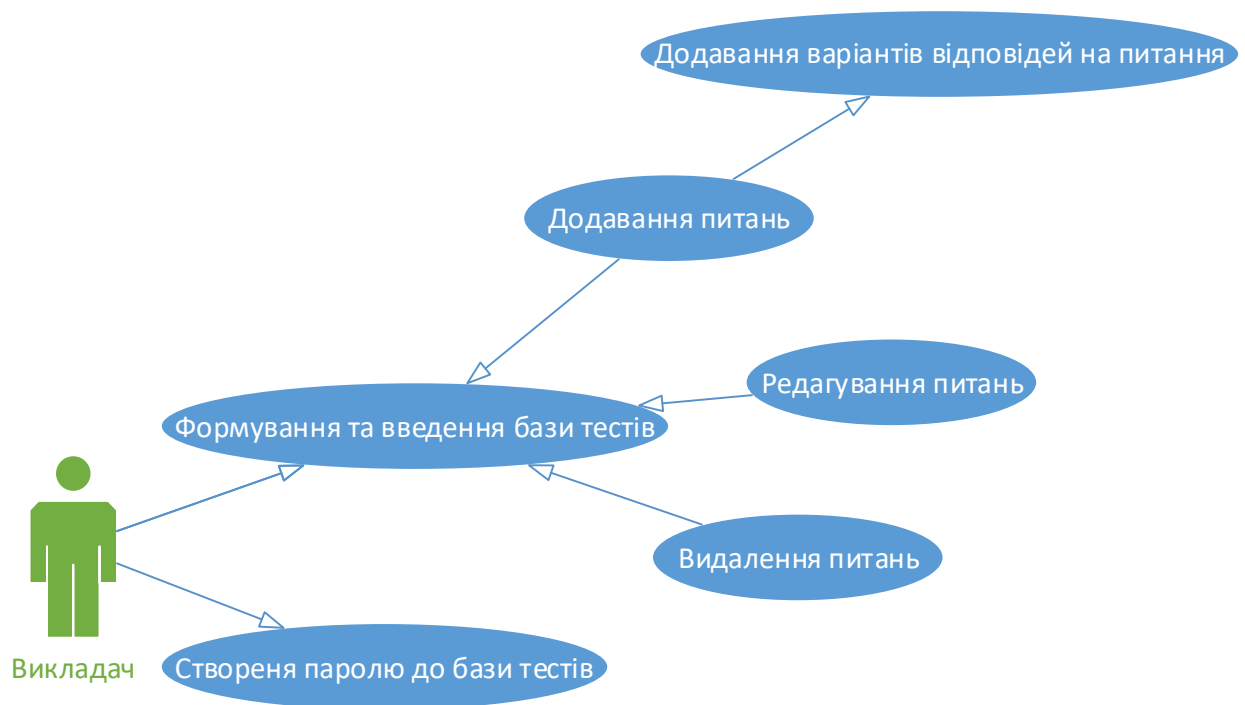


Рисунок 1.2. Варіанти використання ПЗ для викладача

Опис прецедентів використання для актора «Студент» (рис. 1.1).

1. Прецедент «Реєстрація користувача».

Прецедент активізується актором «Студент» на початку проведення тестування. При виконанні прецеденту система надає форму для введення ПІБ користувача з кнопкою для підтвердження і переходу до тестування.

2. Прецедент «Вибір чергового (відкладеного) питання тесту».

Користувач може обрати необхідне йому питання зі списку або по номеру питання та система перейде до необхідного питання.

3. Потік подій для прецеденту «Формування відповіді»:

Головний потік активізується актором «Студент», система надає 2 можливі варіанти дій: вибір та підтвердження відповіді, пропуск питання:

- при виборі опції «Вибір та підтвердження відповіді», користувач обирає відповідь або декілька відповідей на питання, система зберігає вибрану відповідь та відображає стан питання, як вирішеного на екрані;

- якщо користувач обирає «Пропуск питання», то система відкладає питання та помічає його, як питання без відповіді. Після цього обирається наступне питання в тестуванні.

4. Прецедент «Формування та виведення результатів тестування».

Прецедент активізується актором «Студент».

Головний потік – система надає можливість завершити тестування та отримати результати в екранній або друкованій формі.

Постумови. Сформовано файл з результатами тестування у вигляді документа PDF.

Опис прецедентів використання для актора «Викладач» (рис. 1.2).

1. Прецедент «Формування та ведення бази тестів».

Прецедент активізується актором «Викладач» і надає можливі варіанти дій: додавання питання до бази тестів, з варіантами відповідей на питання; редагування питань бази тестів; видалення питань:

- якщо обрана операція додавання питання, то користувач отримує форму з полями для введення питання та варіантів відповідей, сформований запис додається до бази тестів;
- якщо обрана операція редагування питання, то надається форма з інструментами пошуку необхідного питання; обране питання виводиться на форму у відповідні поля; користувач вносить корективи та обирає режим збереження; скоригований запис зберігається в базі тестів;
- якщо обрана операція видалення питання, то надається форма з інструментами пошуку необхідного питання; обране питання видаляється з бази тестів.

Постумови. Новий або скоригований запис в базі даних.

2. Прецедент «Створення паролю до бази тестів». Користувач може створити пароль для захисту бази тестів.

Дослідження потоків даних виконано за допомогою діаграми потоків даних. Діаграма потоків даних дозволяє виявити сутності, що приймають участь у функціонуванні системи, процеси, які відбуваються та інформацію, яка змінюється в процесі роботи програмного забезпечення. На рисунку 3 зображена діаграма потоків даних нульового рівня.

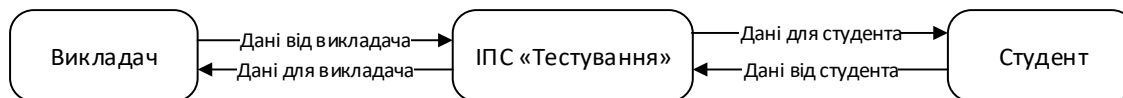


Рисунок 1.3. Діаграма потоків даних нульового рівня

На діаграмі наявні 2 зовнішні сутності «Викладач» та «Студент».

Дані від викладача – завдання, тести.

Дані для викладача – файл результату та звіт проходження тестування.

Дані від студента – відповіді на завдання тесту.

Дані для студента – результати проходження тесту.

На рис. 1.4 наведено декомпозицію процесу комп'ютерного тестування.

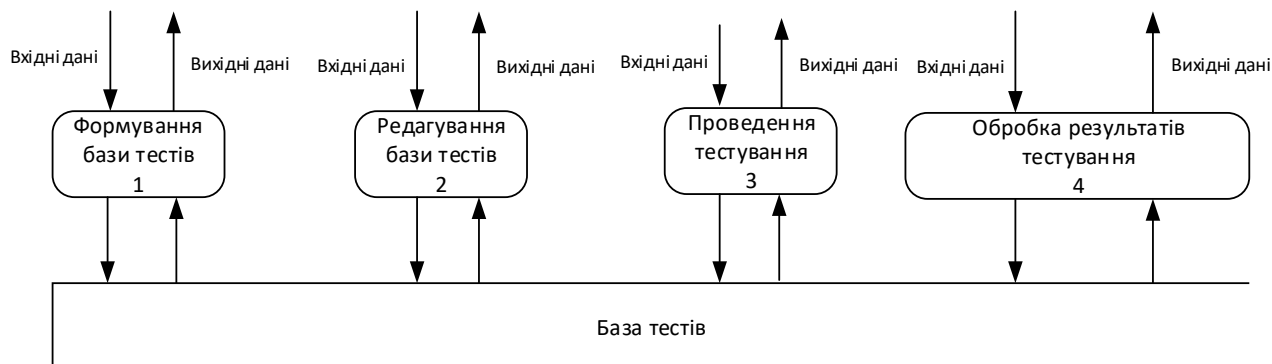


Рисунок 2.4. Діаграма потоків даних першого рівня

В результаті декомпозиції виділено чотири підпроцеси. Опис потоків даних наведено в табл. 1.1.

Таблиця 1.1

Словник структури потоків даних діаграми першого рівня

Потоки даних діаграми 0 рівня	Потоки даних діаграми 1 рівня
Дані від викладача	Вхідна інформація
Дані для викладача	Вихідна інформація
Дані від студента	Вхідна інформація
Дані для студента	Вихідна інформація

На рис. 1.5 наведено деталізацію процесу «Формування бази тестів». В табл. 1.2 наведено словник потоків даних діаграми потоків даних другого рівня процесу «Формування бази тестів».

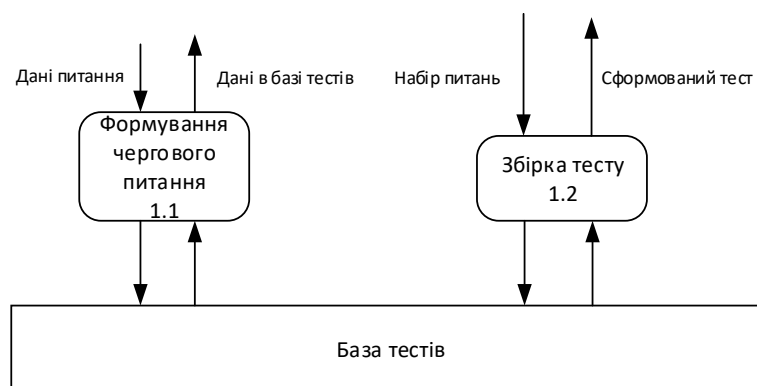


Рисунок 1.5. Діаграма потоків даних другого рівня процесу «Формування бази тестів»

Таблиця 1.2

Словник потоків даних діаграми потоків даних другого рівня процесу «Формування бази тестів»

Потоки даних діаграми 1 рівня	Потоки даних діаграми 2 рівня	Атрибути
Вхідні дані	Дані питання	Текст питання, варіанти відповіді, індекс тесту, номер питання
Вихідні дані	Сформований тест	Об'єкт даних, що містить тест

Потоки даних приведені до рівня атрибутів. Визначено підпроцеси, які не потребують подальшої декомпозиції, так як являють собою елементарні операції (їх можна реалізувати окремими модулями).

Процес «Редагування бази тестів» є також складним, тому його необхідно деталізувати у вигляді діаграми потоків даних другого рівня (рис. 1.6). В табл. 1.3 наведено словник потоків даних діаграми потоків даних другого рівня процесу «Редагування бази тестів».



Рисунок 1.6. Діаграма потоків даних другого рівня процесу «Редагування бази тестів»

Таблиця 1.3

Словник потоків даних діаграми потоків даних другого рівня процесу «Редагування бази тестів»

Потоки даних діаграми 1 рівня	Потоки даних діаграми 2 рівня	Атрибути
Вхідні дані	Корективи питання	Текст питання, варіанти відповіді, індекс тесту, номер питання
	Відредаговані дані тесту	Текст питання, варіанти відповіді, індекс тесту, номер питання
Вихідні дані	Скориговане питання	Відредагований об'єкт даних, що містить текст питання та варіанти відповідей
	Скоригований тест	Об'єкт даних, що містить скоригований тест

Процес «Проведення тестування» забезпечує тестування, яке полягає в наданні чергового питання, прийняття відповіді студента. В подальшому відповіді підлягають аналізу та формуванню. За результатами аналізу формується підсумковий бал. Вхідними даними процесу є одна чи декілька відповідей (в залежності від типу тесту), які вибирає користувач на поточне питання тесту. Вихідними даними є дані відповідей на тест, які подані у вигляді колекції об'єктів даних.

На рис. 1.7 зображена деталізація процесу «Обробка результатів тестування» у вигляді діаграми потоків даних другого рівня ПЗ «Обробка результатів тестування». В табл. 1.4 наведено словник потоків даних діаграми потоків даних другого рівня процесу «Обробка результатів тестування».

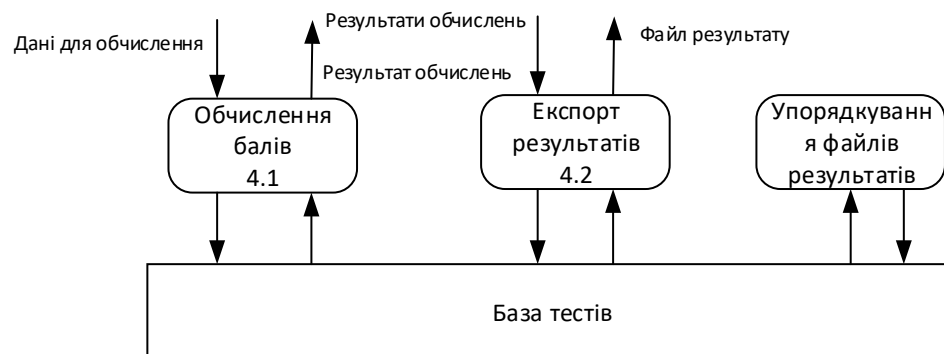


Рисунок 1.7. Діаграма потоків даних другого рівня процесу «Обробка результатів тестування»

Таблиця 1.4

Словник потоків даних діаграми потоків даних другого рівня процесу «Обробка результатів тестування».

Потоки даних діаграми 1 рівня	Потоки даних діаграми 2 рівня	Дані
1	2	3
Вхідні дані	Дані для обчислень	Колекція об'єктів, що містять дані відповідей на питання тесту

1	2	3
	Результати обчислень	Об'єкт даних, що містить кількість балів, перелік вірних та невірних відповідей.
Вихідні дані	Результат обчислень	Об'єкт даних, що містить кількість балів, перелік вірних та невірних відповідей.
	Файл результату	Файл, що містить результати обчислень та звіт

Дослідження потоків даних дозволило побудувати концептуальну модель предметної галузі. Концептуальна модель – це абстрактна модель, що виявляє причинно-наслідкові зв'язки, властиві об'єкту, що досліджується, і суттєві в рамках певного дослідження. Основне призначення концептуальної моделі – виявлення набору причинно-наслідкових зв'язків, облік яких необхідний для отримання необхідних результатів. Вона представляє загальний погляд на дані. Концептуальна модель, яка зображена на рис. 1.8 містить 4 об'єкта та відображає наступні зв'язки між ними:

1. База тестів → Включає → Питання
2. Питання → Включає → Відповідь
3. Питання → Формує → Результати тестування

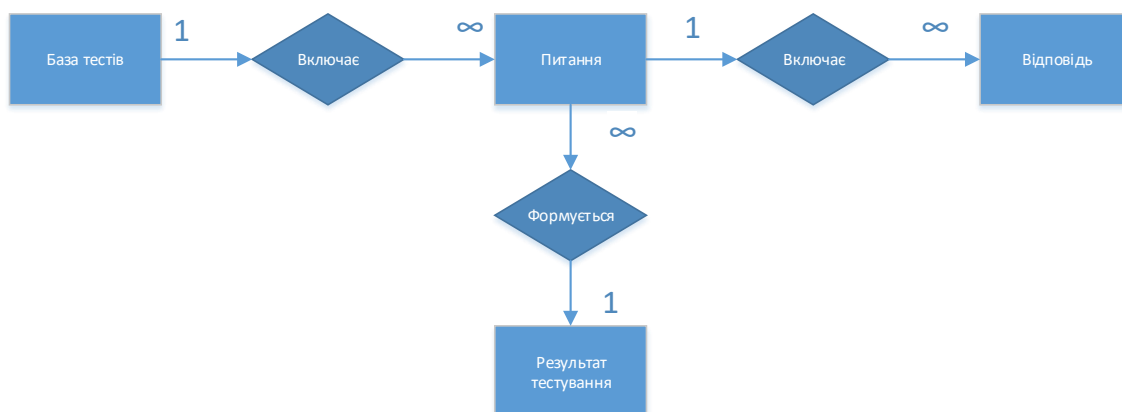


Рисунок 1.8. Концептуальна модель даних

На основі моделі потоків даних та концептуальної моделі даних ми побудували модель переходів програми у вигляді діаграми станів. Діаграма станів програми представляє її динамічну поведінку в результаті зовнішніх дій користувача або модулів програми (рис. 1.9).

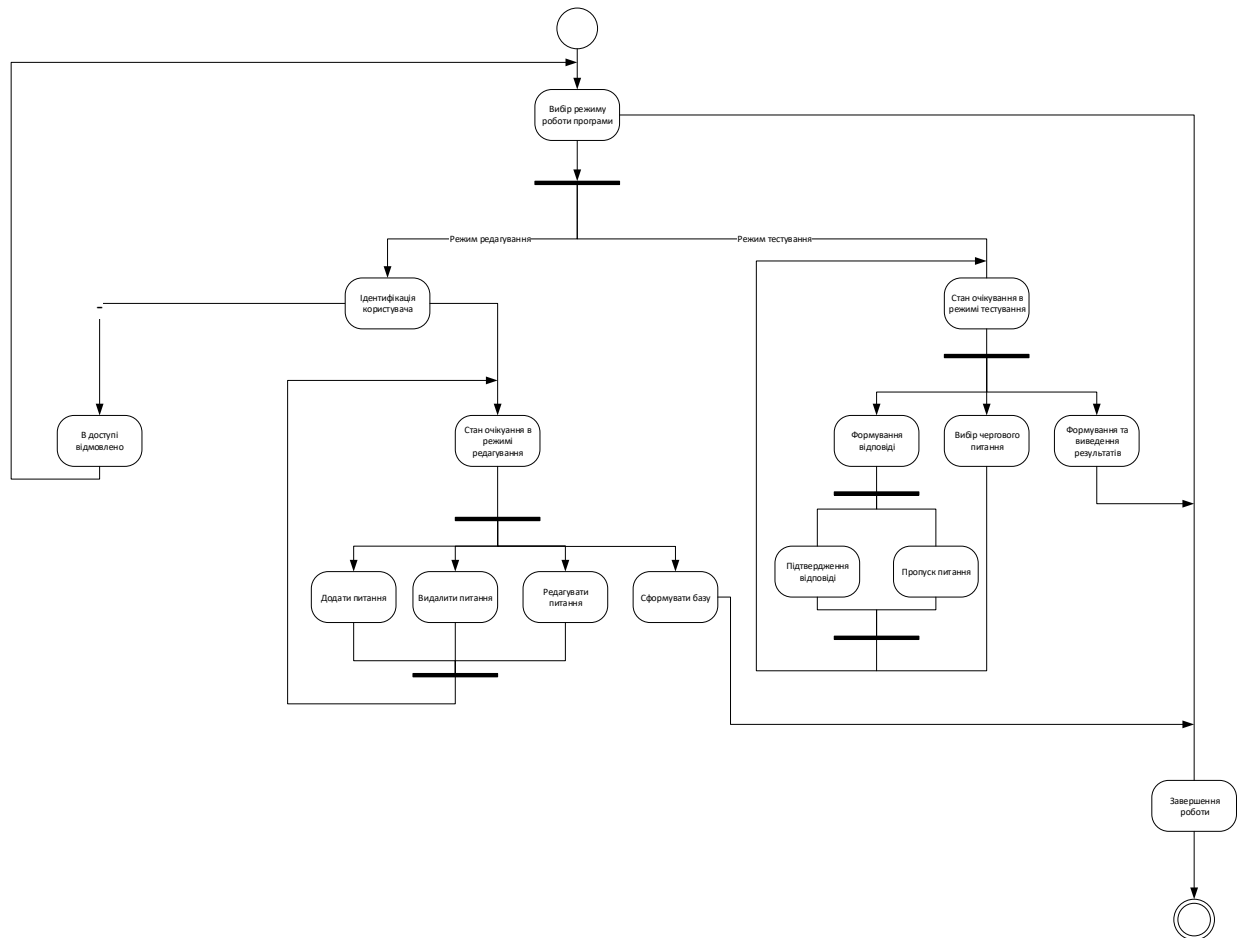


Рисунок 1.9. Діаграма станів програми

Основні стани, в яких може перебувати програма наступні: стан ідентифікації користувача, стан очікування в режимі редагування, стан очікування в режимі тестування та стан завершення роботи. В стані ідентифікації користувача перевіряються права доступу до бази тестів. Якщо доступ отримано програма переходить в стан редагування бази тестів, інакше користувачу буде відмовлено у доступі, і він може перейти у стан тестування або в стан завершення роботи. Стан очікування в режимі редагування надає можливість користувачу редагувати базу тестів або перейти у стан завершення

роботи. Стан очікування в режимі тестування дозволяє користувачу вибирати чергове питання та давати відповіді. З стану очікування можна перейти до стану завершення роботи з відображенням результатів пройденого тестування. Стан завершення роботи зберігає зміни зроблені користувачем та завершує роботу з базою тестів.

На основі цього можна побудувати логічну модель даних, що визначає зв'язки між об'єктами у файлі. Файлова система задачі побудована за вимогами ООП [6]. Файли являють собою об'єкти класу. Ієрархія зв'язків зображена на рис. 1.10. Головним об'єктом програмної системи є Test, що містить в собі колекцію (масив) питань. Наступним в ієрархії зв'язків є об'єкт типу Question. Цей об'єкт описує повну структуру питання, а саме текст питання, масив відповідей та кількість відповідей. Масив відповідей складається з об'єктів типу Answer. В свою чергу Answer є батьківським класом для двох наступних класів: AnswerMatch та AnswerText.

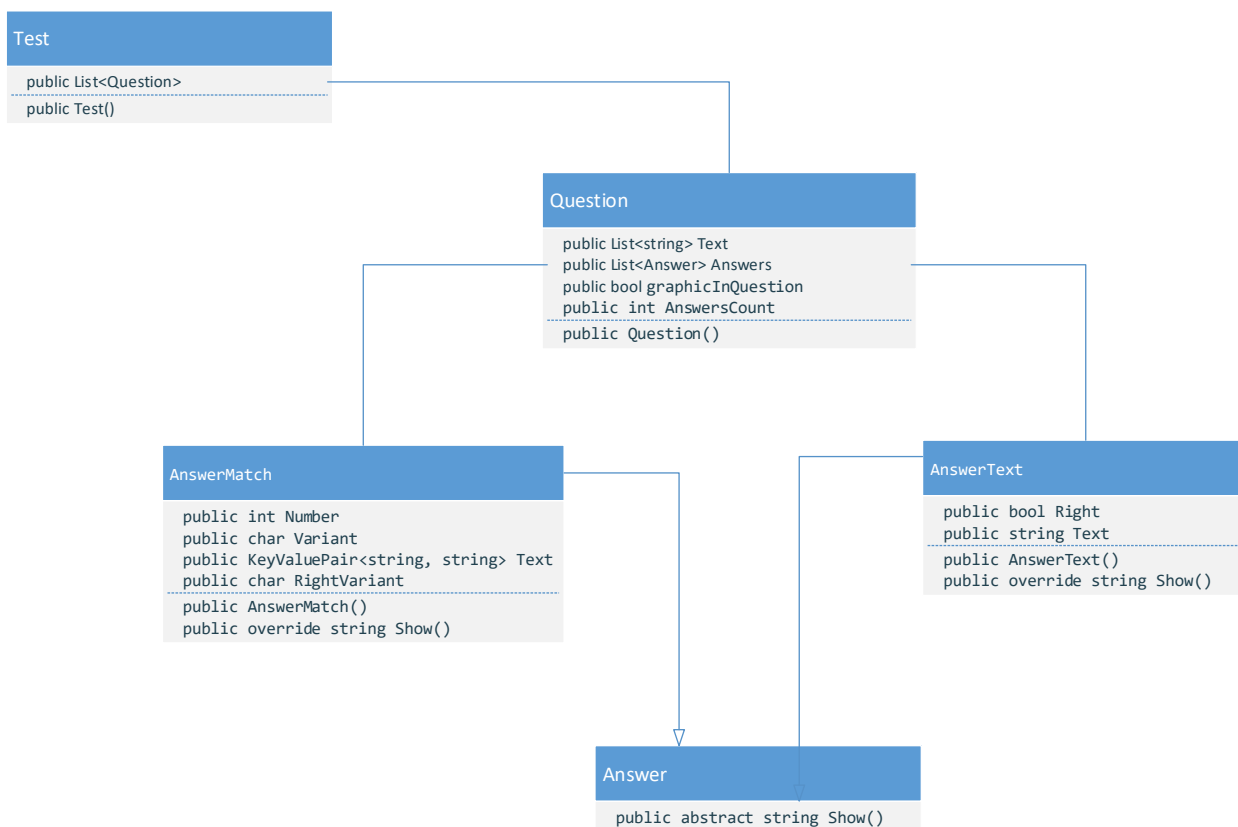


Рисунок 1.10. Діаграма класів

На основі досліджень потоків даних, діаграми варіантів використання та діаграми станів програми ми спроектували архітектуру програмного забезпечення задачі. Для реалізації функцій програми нами використано середовище Visual Studio, мова програмування C#. Структура програми побудовано у вигляді ієрархії класів. Базовий клас містить опис змінних та методів для реалізації визначених функцій. Кожен клас наступного рівня ієрархії реалізує конкретну функцію:

1. формування тесту;
2. редагування тесту;
3. проведення тестування;
4. формування результатів.

Клас який реалізує формування тесту забезпечує створення питань тесту, створення відповідей на питання та збереження питань тесту в базі. Об'єкт бази тесту має структуру, наведену у табл. 1.5.

Таблиця 1.5

Структура об'єкту бази тесту

Логічне ім'я	Тип
Зміст питання	List<string>
Список відповідей	List<Answer>
Запланована кількість відповідей	int
Конструктор питання	-

Так як питання орієнтовані на два типи відповідей, то в програмі необхідно передбачити два типи об'єктів відповідей. Перший тип об'єкту відповіді (AnswerText) на запитання тесту має структуру, наведену у табл. 1.6. Другий тип об'єкту відповіді (AnswerMatch) на запитання тесту має наступну структуру, наведену у табл. 1.7.

Таблиця 1.6

Структура об'єкту відповіді

Логічне ім'я	Тип
Ознака правильної відповіді	bool
Текст відповіді	string
Конструктор відповіді	-
Метод відображення вмісту відповіді	string

Таблиця 1.7

Структура об'єкту відповіді

Логічне ім'я	Тип
Номер відповіді	int
Позначення відповіді	char
Питання-відповідь	Структура(string, string)
Вірний варіант	char
Конструктор відповіді	-
Метод відображення вмісту відповіді	string

Об'єкти першого і другого типу в свою чергу є нащадками класу Answer. Абстрактний клас Answer оголошує метод відображення вмісту.

Специфікація модулів ПЗ мають наступну специфіку.

1. Модуль (клас) Test реалізує формування та ведення бази тестів:

Вхідна інформація – вибір режиму роботи.

Призначення модуля – оголошення масиву питань і перехід до вибраного режиму роботи.

Вихідна інформація – масив питань тесту.

2. Модуль (клас) Manager виступає головною керуючою структурою проекту

Вхідна інформація – об'єкт типу Test.

Призначення модуля – реалізує зв'язок між всіма модулями програми.

Вихідна інформація – об'єкт типу Test.

3. Модуль (клас) ReadWriteTest реалізує методи зчитування та запис бази тестів у пам'ять комп'ютера.

Вхідна інформація – файл бази тестів.

Призначення модуля – зчитування вмісту файлу бази тестів та перетворення у об'єкт типу Test та навпаки.

Вихідна інформація – об'єкт типу Test.

4. Модуль (клас) MainForm реалізує інтерфейс користувача програмного забезпечення.

Вхідна інформація – команди користувача.

Призначення модуля – створення графічного інтерфейсу управління програмним продуктом, зчитування та відображення результатів роботи функцій програми.

Вихідна інформація – результати виконання команд.

Для відображення взаємодії об'єктів, впорядкованих за часом, було побудовано діаграму послідовності. При виборі режиму (1, 12) завантажується відповідний модуль програми – створення та редагування тесту (1) або проходження тесту (12). Редагування тесту складається з створення питань тесту (2), при додаванні питання для переходу до його редагування обирається одна з функцій для редагування (3-8). Після формування питань встановлюється пароль до бази тестів (9). Сформована база тестів зберігається в пам'яті комп'ютера (10-11). При виборі режиму проходження тесту користувач

відповідає на питання тесту та переходить по питанням (13), після чого, давши відповіді на всі запитання, користувач завершує тест та бачить свій результат у вікні програми та в згенерованому файлі з розширенням PDF (14).

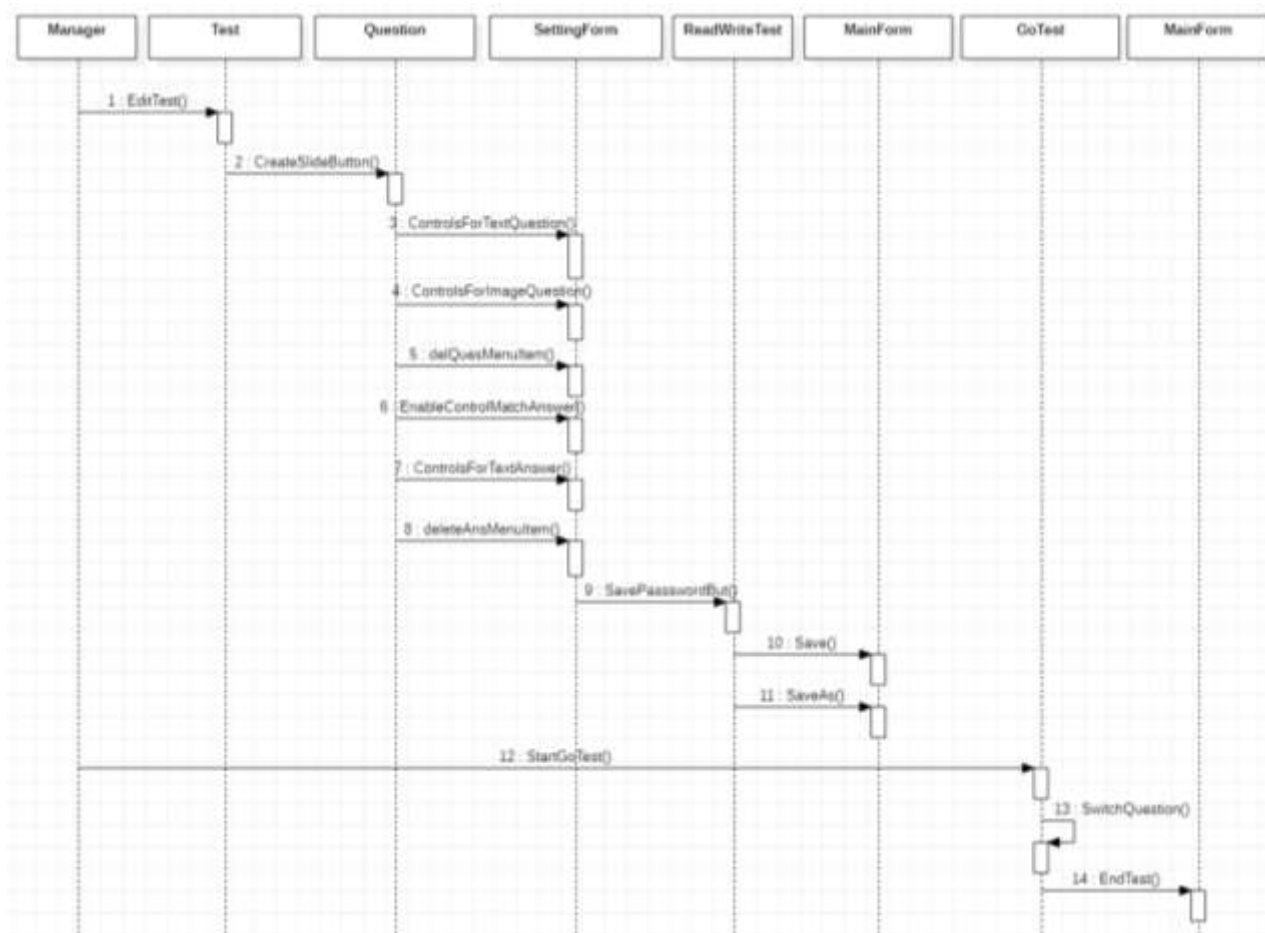


Рисунок 1.11. Діаграма послідовності

Отже, нами було розглянуто основні стани програмного продукту на всіх стадіях життєвого циклу. Описано варіанти використання програмного забезпечення, проведена деталізація процесу за допомогою діаграми потоків даних, описані причинно-наслідкові зв'язки у вигляді концептуальної моделі переходів програми, та спроектовано архітектуру програмного забезпечення.

РОЗДІЛ 2. ІНТЕРФЕЙС ТА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ ТЕСТУВАННЯ ЗНАНЬ

Комп'ютерну систему тестування знань реалізовано як десктоп-застосунок з графічним інтерфейсом користувача, який має віконну систему. При старті програми відкривається вікно вибору режиму роботи (рис. 2.1).

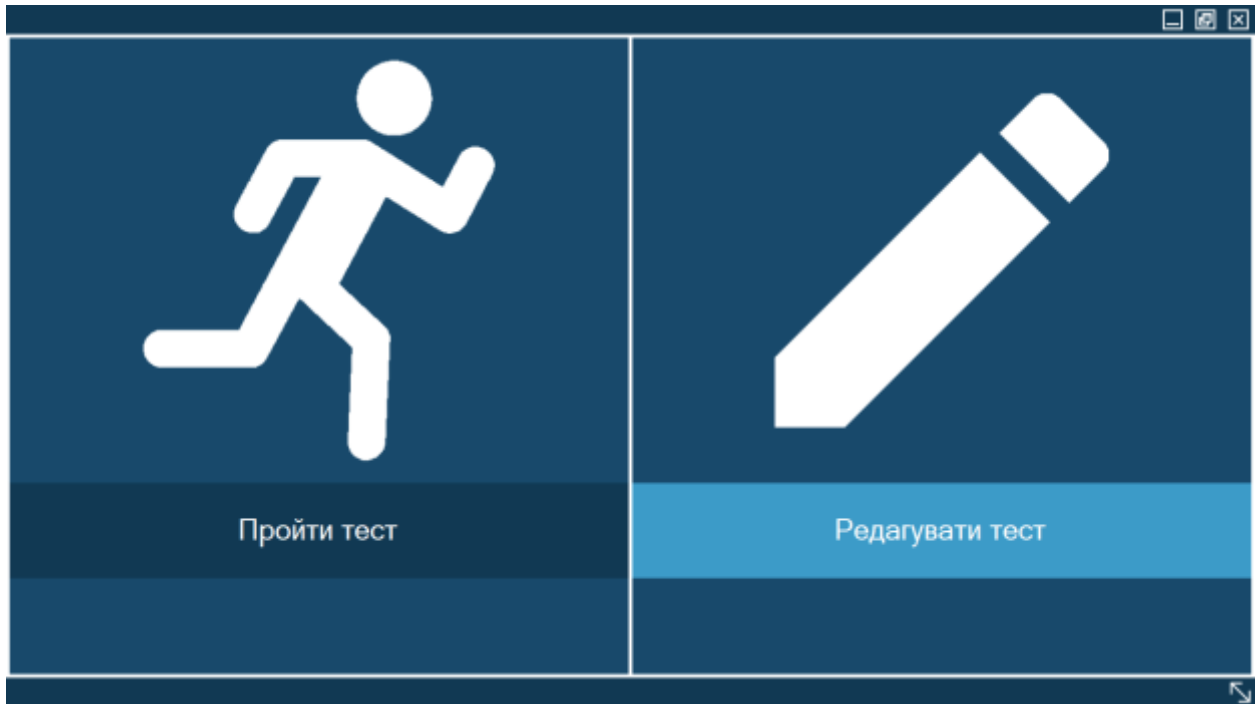


Рисунок 2.1. Вибір режиму роботи програми

Далі наведено інструкції користувача в режимі редагування бази тестів.

Послідовність дій для створення нового файлу тестів:

- запустити програму за допомогою ярлику;
- у лівому боковому меню обрати пункт «Створити» (рис. 2.2).

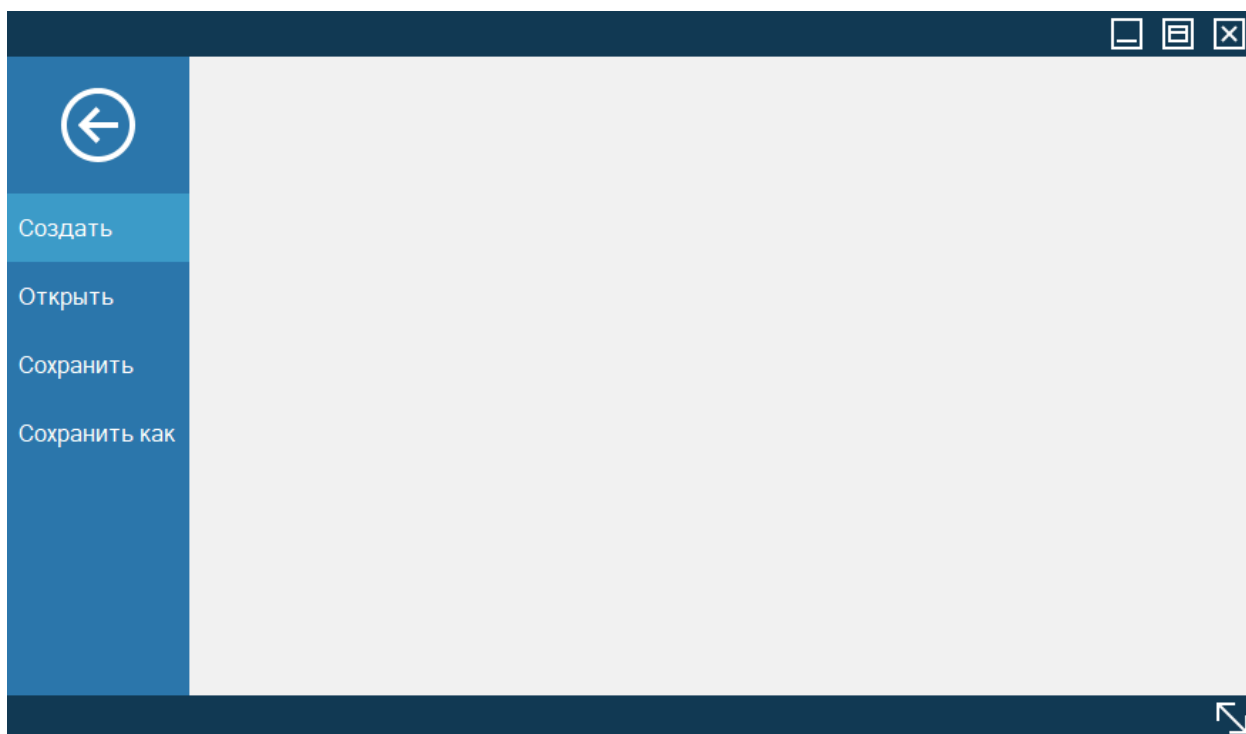


Рисунок 2.2. Створення бази тестів

Послідовність дій для збереження файлу тестів:

- на стрічці меню обрати команду «Файл»;
- у випадаючому меню управління файлами обрати команду «Зберегти як»;
- у діалоговому вікні «Select a Cursor File» обрати шлях для збереження файлу (рис. 2.3).

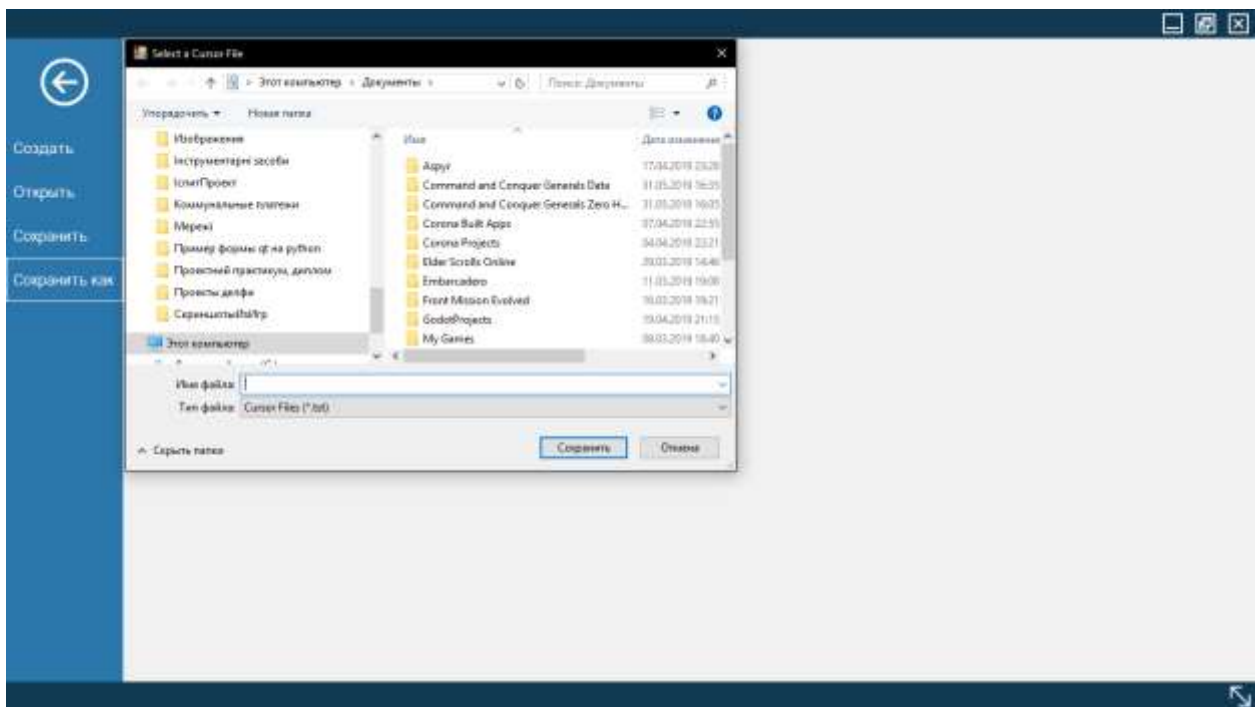


Рисунок 2.3. Збереження бази тестів

Послідовність дій для створення нового текстового питання:

- на стрічці меню обрати команду «Вставка»;
- натиснути на кнопку «Додати питання»;
- обрати «Текст» (рис. 2.4).

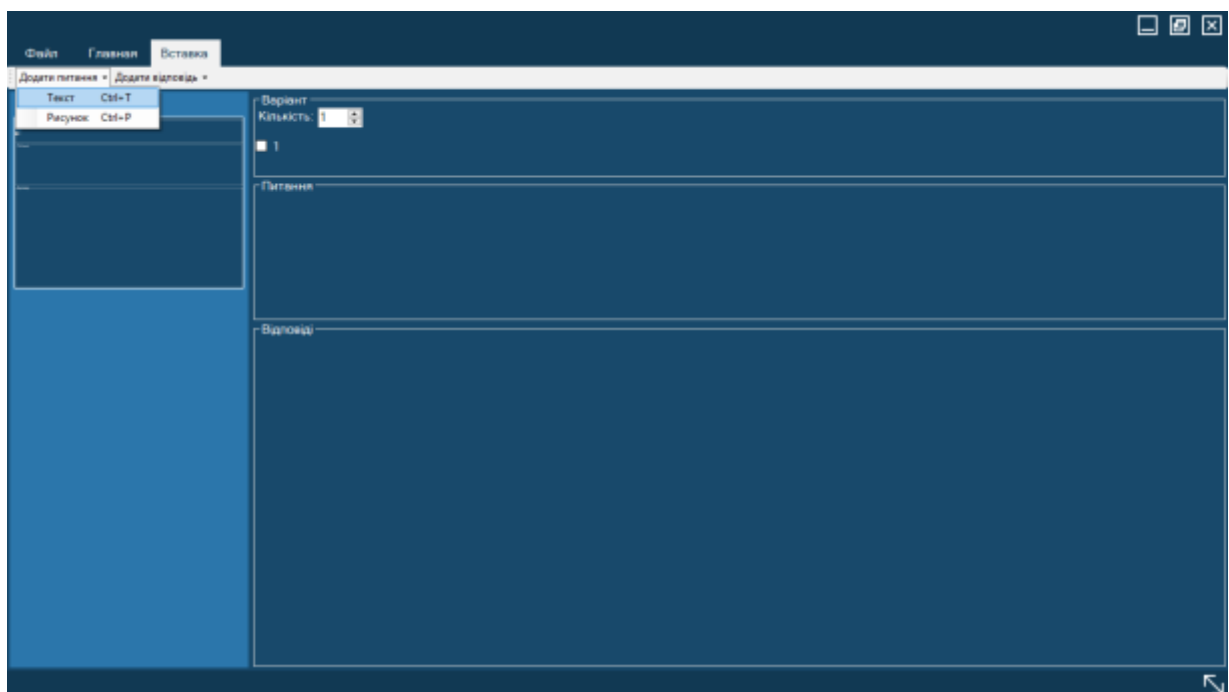


Рисунок 2.4. Створення текстового питання

Послідовність дій для створення нового графічного питання:

- на стрічці меню обрати команду «Вставка»;
- натиснути на кнопку «Додати питання»;
- обрати «Рисунок» (рис. 2.5).

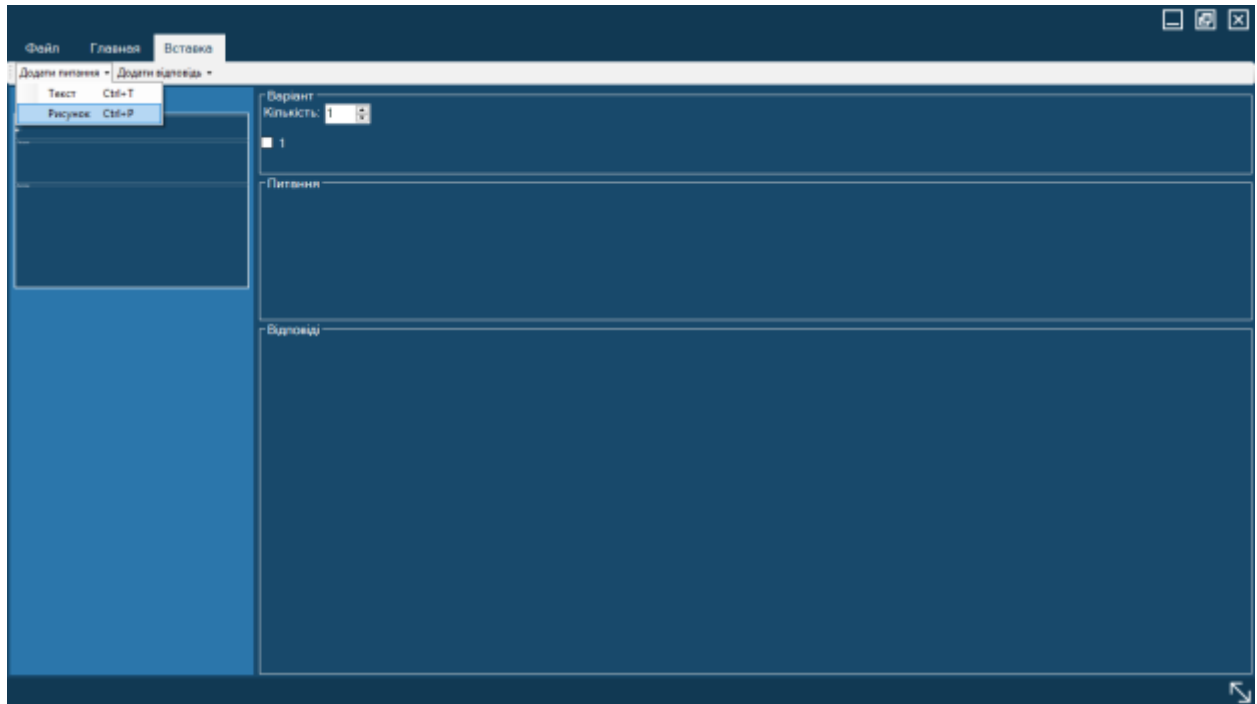


Рисунок 2.5. Створення графічного питання

Послідовність дій для створення текстової відповіді на питання:

- на лівій боковій стрічці вибрати необхідне питання;
- на стрічці меню обрати команду «Вставка»;
- натиснути на кнопку «Додати відповідь»;
- обрати «Варіант» (рис. 2.6).

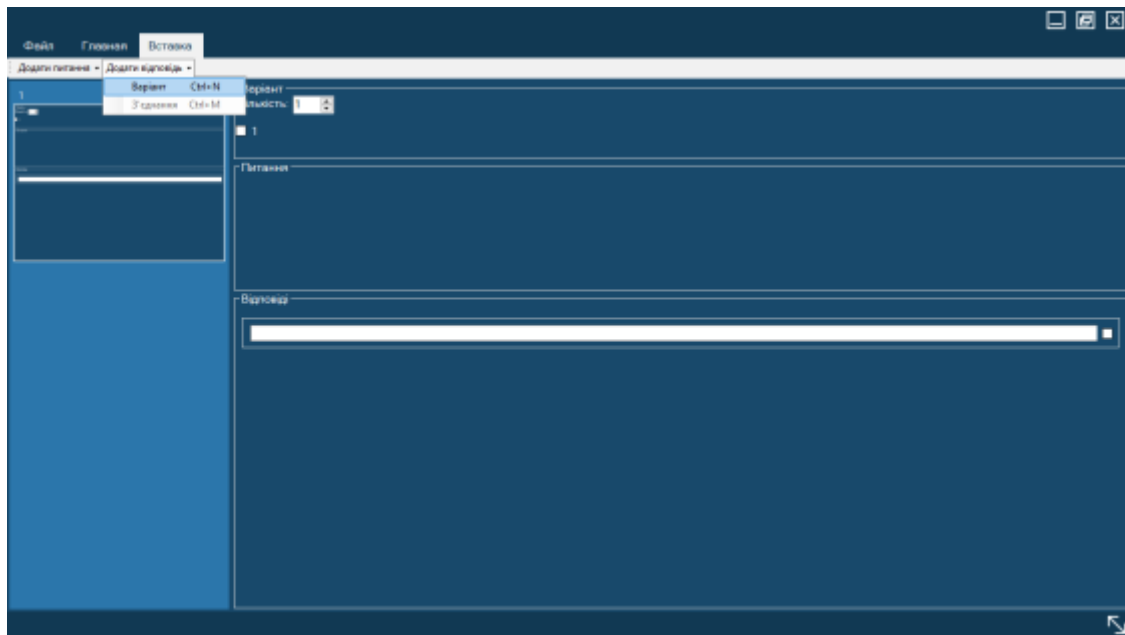


Рисунок 2.6. Створення текстової відповіді

Послідовність дій для фіксації відповіді, як правильної:

- на лівій боковій стрічці вибрати необхідне питання;
- вибрати існуючу відповідь на питання (рис. 2.7).

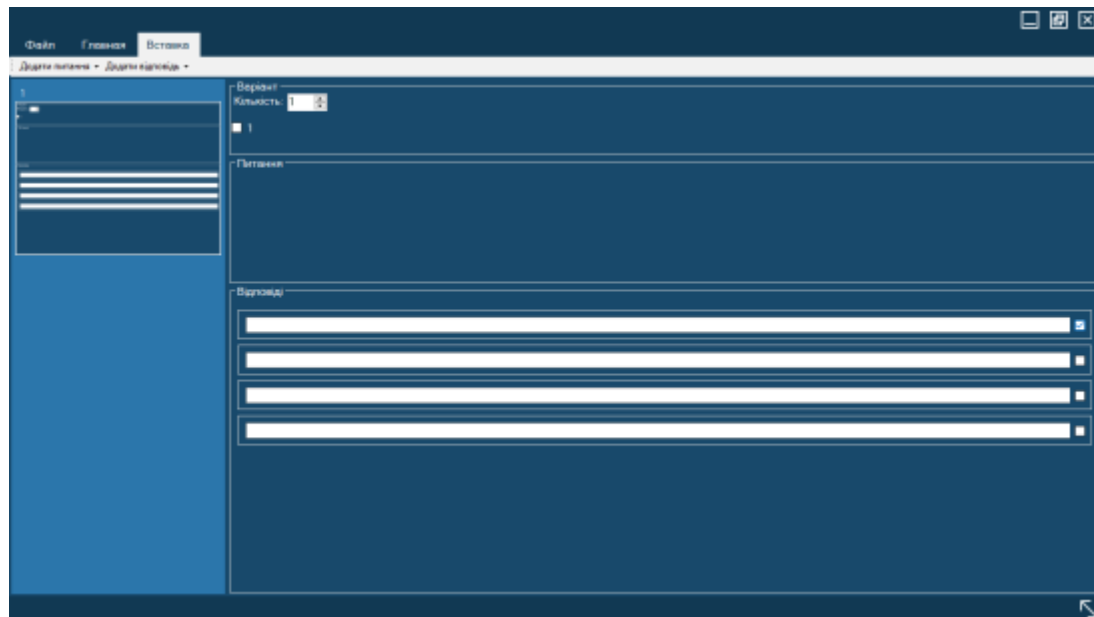


Рисунок 2.7. Фіксація правильної відповіді

Послідовність дій для створення відповіді з'єднання:

- на лівій боковій стрічці вибрати необхідне питання;
- на стрічці меню обрати команду «Вставка»;

- натиснути на кнопку «Додати відповідь»;
- обрати «З'єднання» (рис. 2.8).



Рисунок 2.8. Створення відповіді з'єднання

Для завершення роботи з програмою в правому верхньому кутку натиснути кнопку «Завершити» (рис. 2.9).

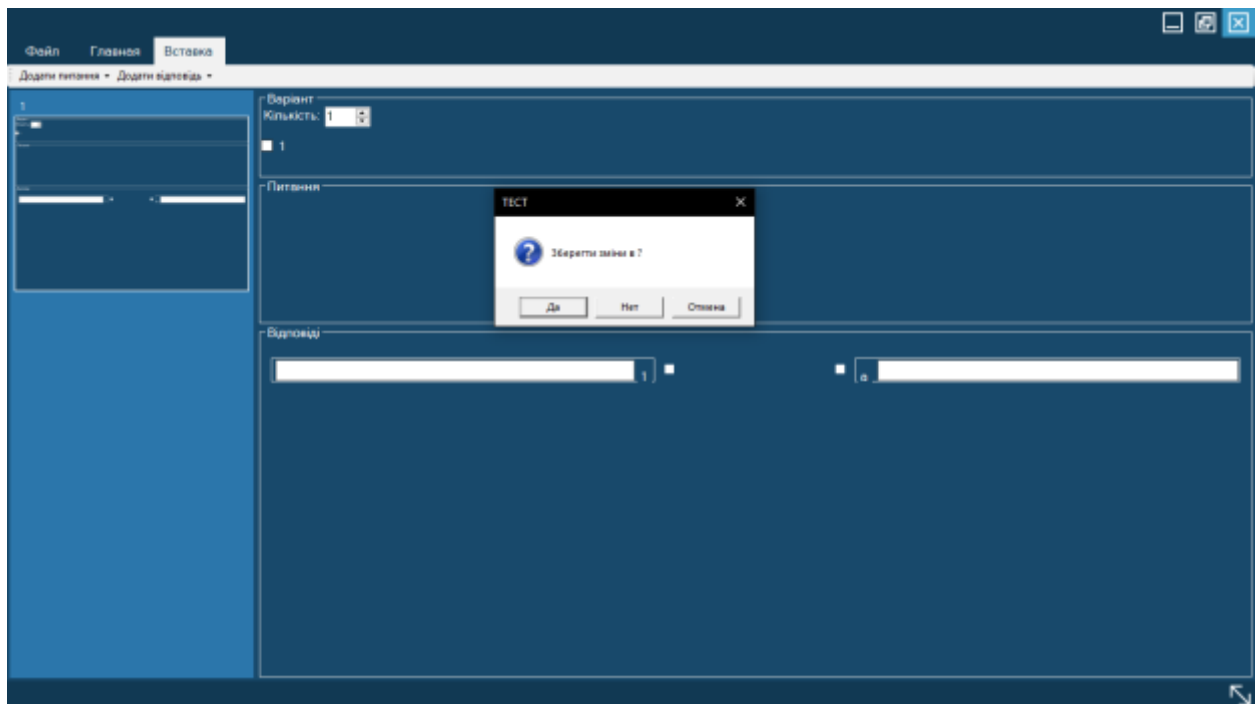


Рисунок 2.9. Завершення роботи з програмою

Далі наведено опис модуля проходження тесту.

Перед проходженням тесту необхідно обрати варіант та ввести ПІБ (рис. 2.10).

The screenshot shows a dark blue window with a title bar. On the left, there is a vertical list of three options, each with a number and a horizontal line: '1', '2', and '3'. To the right of this list is a label 'Варіант' (Variant). Further right is a dropdown menu currently showing the number '1'. Below the variant selection is a white icon of a person, followed by the label 'ПІБ' (Full Name). To the right of the icon is a long, empty white text input field. At the bottom center of the window is a large, dark blue button with the word 'Старт' (Start) in white text. The window has standard OS window controls (minimize, maximize, close) in the top right corner.

Рисунок 2.10. Введення даних для проходження тесту

Для відповіді на завдання необхідно клацнути на відповідному прапорцю (рис. 2.11).

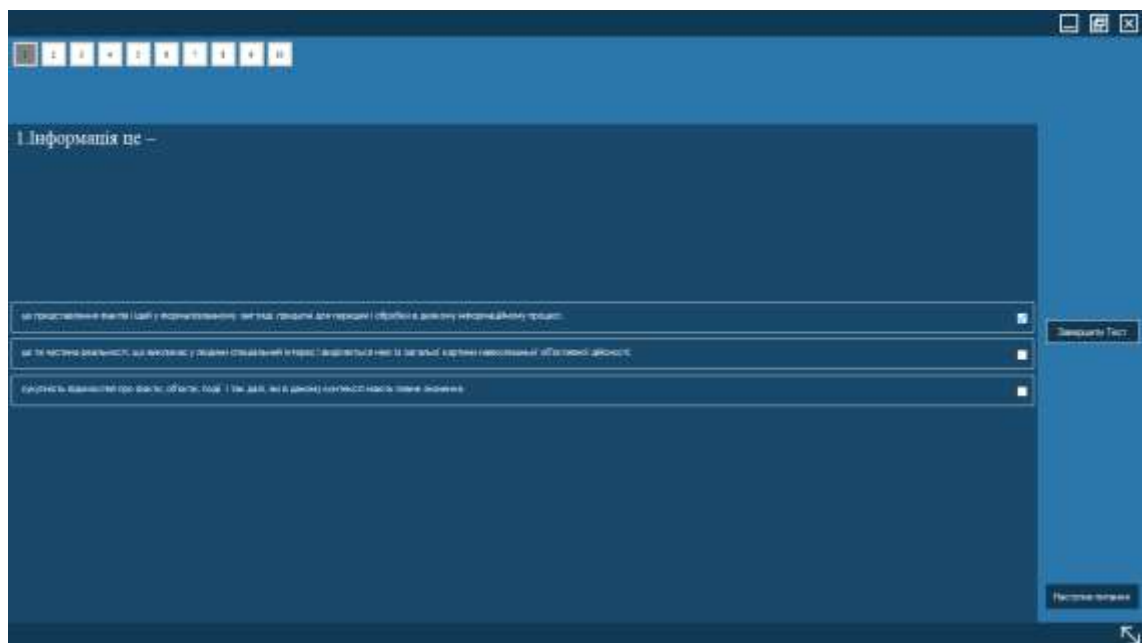
The screenshot shows a dark blue window with a title bar. At the top left, there is a row of seven small, light blue square buttons. Below this is a section titled 'Інформація про...' (Information about...). Underneath the title, there are three rows of text, each followed by a small square checkbox. The first row of text is: 'На представленні варіанти і дані у відповідності з питанням, потрібно обрати один з варіантів інформації, що є правильним.' The second row of text is: 'На дані питання, потрібно обрати один з варіантів інформації, що є правильним.' The third row of text is: 'Правильність відповіді на питання, що є правильним, буде визначена після закінчення тесту.' On the right side of the window, there are two buttons: 'Закінчити Тест' (End Test) and 'Наступне питання' (Next Question). The window has standard OS window controls in the top right corner.

Рисунок 2.11. Вибір відповіді на завдання

Для переходу на наступне питання необхідно натиснути на кнопку «Наступне питання» або натиснути на номер питання у верхній частині вікна (рис. 2.12).

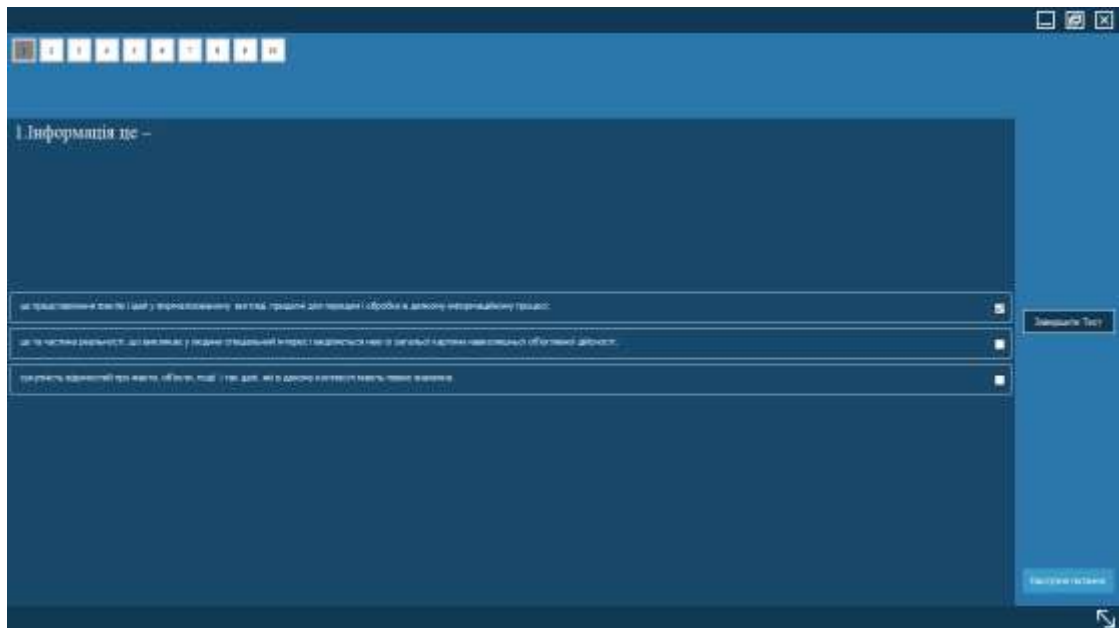


Рисунок 2.12. Вибір наступного завдання

Для завершення тестування необхідно натиснути на кнопку «Завершити тест». Програма завершить тестування та виведе результати (рис. 2.13-2.14).

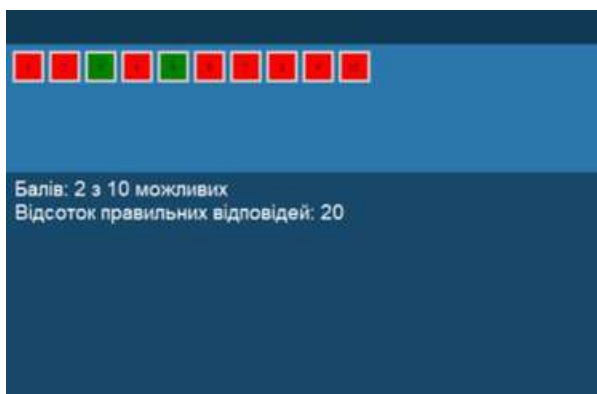


Рисунок 2.13. Результати тестування

Протокол тестування.
Здобувач освіти: Кіяшко Максим Сергійович

Дисципліна	Правильний варіант	Вибраний варіант
Відсутній	1 2 3 1 3 1 2 3 1	1 3 3 2 1 2 3 1 3
		Відсоток правильних відповідей: 0 %
Матем		Відсоток правильних відповідей: 100 %
Алгоритми	2	2
		Відсоток правильних відповідей: 100 %
		Загальний відсоток правильних відповідей: 66,6666 %

Рисунок 2.14. Результати на друк

ВИСНОВКИ

Нами було розроблено автоматизовану систему тестування знань, що дозволяє ефективно та швидко проводити оцінку знань.

Розроблена система тестування знань має такі можливості:

- створення тесту з питаннями (1 тест – 1 файл з розширенням .tst);
- редагування тесту;
- гнучке налаштування тестів;
- збереження тесту;
- захист режиму проектування тесту паролем;
- проходження тесту;
- форматоване виведення результатів тестування на друк.

Програмний продукт надає більшу точність та швидкість в процесі проведення та проектування тестування, ніж при ручному тестуванні. Процес проектування тестів є універсальним та користувач має змогу компонувати окремі питання в тестах між собою, швидко редагувати питання тесту, конвертувати тести з формату .doc та .docx.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Singleton паттерн [Електронний ресурс] : MetaInt, 2018. – Режим доступу : URL : <https://metanit.com/sharp/patterns/2.3.php>. – Загол. з екрану.
2. C# Windows Forms tutorial [Електронний ресурс] : ZetCode, 2019. – Режим доступу : URL: <http://zetcode.com/csharp/windowsforms/>. – Загол. з екрану.
3. Основи Windows Forms. [Електронний ресурс] : Уроки OpenGL + C#, 2018. – Режим доступу : URL: <http://esate.ru/uroki/OpenGL/uroki-OpenGL-c-sharp/osnovi-windows-forms/>. – Загол. з екрану.
4. Документація по C# [Електронний ресурс] : Microsoft Docs, 2019. – Режим доступу : URL: <https://docs.microsoft.com/ru-ru/dotnet/csharp/>. – Загол. з екрану.
5. Rob Miles “C# Programming Yellow Book”, 216 p., Year: 2016 ISBN: 9781728724966.
6. Svetlin Nakov, Veselin Kolev “Fundamentals of Computer Programming with C#”, 1122p., Year: 2013 ISBN: 9789544007737.

ПРОГРАМНИЙ КОД

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
using System.Drawing.Drawing2D;
using System.Threading;
using System.Runtime.InteropServices;
using iTextSharp.text.pdf;
using iTextSharp.text;

namespace IsputCSharpWinFormsV2
{
    public partial class MainForm : Form
    {
        /// <summary>
        /// Інформація про тести
        /// </summary>
        int indexQuestion;
        Thread thread;
        Question currentQuestion {
            get
            {
                if (Manager.Instance.CurrentTest.Questions.Count != 0)
                    return Manager.Instance.CurrentTest.Questions[indexQuestion];
                else
                    return null;
            }
        }
        int indexAnswer;
        Control currentPanel;

        /// </summary>
        private int activeAnswer = -1;
        private System.Drawing.Point mouseStart;
        private System.Drawing.Point mouseEnd;
        private bool isStartMatching;
        private int numberMatch;
        private int variantMatch;
        Bitmap defImageSlide;
        TestData test;
        bool openStartPage;

        public MainForm()
        {
            InitializeComponent();

            _arrows = new List<GraphicsPath>();
            openStartPage = true;
            //Program.arg.Add("C:\\Users\\Maxon\\Desktop\\pas.tst");
            for (int i = 0; i < Program.arg.Count; i++)
            {
                if (Program.arg[i].Split('.')[Program.arg[i].Split('.').Length - 1] == "tst")
                {
                    //ReadWriteTest.GetInstance().ReadTest(Program.arg[i]);
                    openStartPage = false;
                }
            }
            if (openStartPage)
                textBoxPassword.Visible = false;

            SetCurrentPanel(tableLayoutPanelStart);

            for (int i = 0; i < Program.arg.Count; i++)
            {

```

```

        if (Program.arg[i].Split('.')[Program.arg[i].Split('.').Length - 1] == "tst")
        {
            ReadWriteTest.GetInstance().ReadTest(Program.arg[i]);
        }
    }

    //tableLayoutPanelStart.Hide();
    //panelStartGoTest.Visible = true;
    //panelStartGoTest.BringToFront();
    comboBoxSubject.Items.Add(Subject.Відсутній);
    comboBoxSubject.Items.Add(Subject.Алгоритм);
    comboBoxSubject.Items.Add(Subject.Матем);
}

public void SetCurrentPanel(Control panel)
{
    if (currentPanel != null)
        currentPanel.Visible = false;
    currentPanel = panel;
    currentPanel.Visible = true;
}

public void EditTest()
{
    if (textBoxPassword.Text != Manager.Instance.CurrentTest.password && !openStartPage)
    {
        MessageBox.Show("Неправильний пароль", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
    {
        SetCurrentPanel(MainWorkSpacePanel);

        timerUpdateElements.Start();
        indexAnswer = 0;
        indexQuestion = 0;
        MouseMove += MainForm_MouseMove;
        MouseUp += MainForm_MouseUp;
        MouseDown += MainForm_MouseDown;
        Paint += MainForm_Paint;
        isStartMatching = false;

        MouseClick += (senders, evs) => UpdateSlideButton();

        this.FormBorderStyle = FormBorderStyle.None;
        this.DoubleBuffered = true;
        this.SetStyle(ControlStyles.ResizeRedraw, true);

        LeftMenuAndInfPanel.Visible = false;
        MainWorkSpacePanel.Visible = true;

        ///удалит!!!
        //MainTestForm form = new MainTestForm();
        //form.Show();

        //ReadWriteTest.GetInstance().WriteTest();

        //test = new TestData();
        defImageSlide = new Bitmap(panel2.Width, panel2.Height);
        panel2.DrawToBitmap(defImageSlide, new System.Drawing.Rectangle(0, 0, panel2.Width, panel2.Height));

        //Thread read = new Thread(KeyRead);
        //read.Start();

        //test.Show();

        for (int i = 0; i < 25; i++)
        {
            indexAnswer = i;
            ControlsForTextAnswer();
            panel3.Controls[0].Visible = false;
        }
    }
}

```

```

        TableLayoutPanelForMatchAnswer();
        for (int i = 0; i < 25; i++)
        {
            indexAnswer = i;
            ControlsForMatchAnswer();
        }
        //CreateSlideButton();
        //SlideGetFocus(this.SlidesPanel.Controls["0"]);

        //bool openStartPage = true;
        //Program.arg.Add("C:\\Users\\Maxon\\Desktop\\тесты\\tests.tst");
        openStartPage = true;
        for (int i = 0; i < Program.arg.Count; i++)
        {
            if (Program.arg[i].Split('.')[Program.arg[i].Split('.').Length - 1] == "tst")
            {
                ReadWriteTest.GetInstance().ReadTest(Program.arg[i]);
                currentFileName = Program.arg[i];
                SlidesPanel.Controls.Clear();
                for (int j = 0; j < Manager.Instance.CurrentTest.variantCount; j++)
                {
                    if (j > 0)
                    {
                        CheckBox variant = new CheckBox()
                        {
                            Dock = DockStyle.Left,
                            Text = (j + 1).ToString(),
                            Name = "VariantCheckbox_" + j,
                            Checked = currentQuestion.InVariant[j],
                            AutoSize = true
                        };
                        variant.CheckedChanged += Variant_CheckedChanged;
                        panelCheckBoxVariant.Controls.Add(variant);
                    }
                    else
                        (panelCheckBoxVariant.Controls["VariantCheckbox_" + 0] as CheckBox).Checked = currentQuestion.InVariant[0];
                }
                UIForTets();
                openStartPage = false;
                break;
            }
        }

        if (openStartPage)
        {
            //Manager.Instance.CurrentTest.Questions.Add(new Question());
            //UIForTets();
            CreateSlideButton();
            needSave = true;
            saveAs = true;
        }
        else
        {
            needSave = true;
            saveAs = false;
        }
    }

    private void toolStripButtonSettings_Click(object sender, EventArgs e)
    {
        //SettingsForm settingsForm = new SettingsForm();
        //settingsForm.ShowDialog();
        SetCurrentPanel(panelSetting);
    }

    private void toolStripButtonCreateQuestion_Click(object sender, EventArgs e)
    {
        CreateSlideButton();
    }

    private void UpdateSlideButton()

```

```

{
    //Кнопка-слайдом
    Bitmap printscreen = new Bitmap(panel2.Width, panel2.Height);
    panel2.DrawToBitmap(printscreen, new System.Drawing.Rectangle(0, 0, panel2.Width, panel2.Height));
    SlidesPanel.Controls[indexQuestion].BackgroundImage = printscreen;
}

private void CreateSlideButton()
{
    //Створ нове питання
    Manager.Instance.CurrentTest.Questions.Add(new Question());
    indexQuestion = Manager.Instance.CurrentTest.Questions.Count - 1;
    //Панель з питаннями та відповідями очищується
    if (panel3.Controls[MatchCnt.matchLayoutPanel.ToString()] != null)
    {
        for (int i = 0; i < panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].Controls.Count; i++)
        {
            panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].Controls[i].Visible = false;
        }
        for (int i = 0; i < panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].Controls.Count; i++)
        {
            panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].Controls[i].Visible = false;
        }
        for (int i = 0; i < panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].Controls.Count; i++)
        {
            panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].Controls[i].Visible = false;
        }
    }
    for (int i = 0; i < panel3.Controls.Count; i++)
    {
        panel3.Controls[i].Visible = false;
    }
    //panel3.Controls.Clear();
    textToolStripMenuItem.Enabled = true;
    variantToolStripMenuItem.Enabled = true;
    matchToolStripMenuItem.Enabled = true;
    imageToolStripMenuItem.Enabled = true;

    CreateUIForSlide(Manager.Instance.CurrentTest.Questions.Count);
}

public void CreateUIForSlide(int num)
{
    //int num = num; /*Manager.Instance.CurrentTest.Questions.Count*/;
    //SlidesPanel.Controls.Count - 1;
    Button pictureButton = new Button
    {
        Name = (num - 1).ToString(),
        BackgroundImage = this.defImageSlide,
        BackgroundImageLayout = ImageLayout.Stretch,
        Dock = DockStyle.Top,
        FlatStyle = FlatStyle.Flat
    };
    pictureButton.ContextMenuStrip = contextMenuStripSlideDel;
    Label questionNumberLabel = new Label
    {
        Name = "questionNumberLabel_" + (num - 1).ToString(),
        Text = num.ToString(),
        Dock = DockStyle.Top,
        ForeColor = Color.White,
        Padding = new Padding(3, 6, 3, 3),
    };
    pictureButton.FlatAppearance.BorderSize = 1;
    pictureButton.FlatAppearance.BorderColor = Color.FromArgb(17, 57, 83);
    pictureButton.MouseEnter += (object sender2, EventArgs e2) =>
    {
        if (Convert.ToInt32((sender2 as Button).Name) != indexQuestion)

```

```

        (sender2 as Button).FlatAppearance.BorderColor = Color.FromArgb(60, 155, 200);
    };
    pictureButton.MouseLeave += delegate (object sender2, EventArgs e2)
    {
        if (Convert.ToInt32((sender2 as Button).Name) != indexQuestion)
            (sender2 as Button).FlatAppearance.BorderColor = Color.FromArgb(17, 57, 83);
    };
    pictureButton.MouseDown += delegate (object sender2, MouseEventArgs e2)
    {
        timerUpdateElements.Stop();
    };
    pictureButton.MouseClick += delegate (object sender2, MouseEventArgs e2)
    {
        SlideGetFocus(sender2);
    };
    SlidesPanel.Controls.Add(pictureButton);
    SlidesPanel.Controls.Add(questionNumberLabel);

    pictureButton.Size = new Size(pictureButton.Size.Width, (int)(pictureButton.Size.Width * 0.75f));

    SlidesPanel.Controls.SetChildIndex(questionNumberLabel, 0);
    SlidesPanel.Controls.SetChildIndex(pictureButton, 0);

    pictureButton.GotFocus += (senderPB, ev) =>
    {
        for (int i = 0; i < SlidesPanel.Controls.Count; i++)
            if (SlidesPanel.Controls[i] is Button)
                (SlidesPanel.Controls[i] as Button).FlatAppearance.BorderColor = Color.FromArgb(17, 57, 83);

        (senderPB as Button).FlatAppearance.BorderColor = Color.White;
        indexQuestion = Convert.ToInt32(((senderPB as Button).Name));
    };
    for (int i = 0; i < Manager.Instance.CurrentTest.variantCount; i++)
    {
        (panelCheckBoxVariant.Controls["VariantCheckbox_" + i] as CheckBox).Checked = currentQuestion.InVariant[i];
    }
    pictureButton.Focus();
}

public void SlideGetFocus(object sender2)
{
    for (int i = 0; i < SlidesPanel.Controls.Count; i++)
        if (SlidesPanel.Controls[i] is Button)
            (SlidesPanel.Controls[i] as Button).FlatAppearance.BorderColor = Color.FromArgb(17, 57, 83);

    (sender2 as Button).FlatAppearance.BorderColor = Color.White;
    indexQuestion = Convert.ToInt32(((sender2 as Button).Name));
    //Панель з питаннями та відповідями очищується
    QuestionGroupBox.Controls.Clear();
    if (panel3.Controls[MatchCnt.matchLayoutPanel.ToString()] != null)
    {
        for (int i = 0; i < panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].
            Controls[MatchCnt.panelMatchleft.ToString()].Controls.Count; i++)
        {
            panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].
                Controls[MatchCnt.panelMatchleft.ToString()].Controls[i].Visible = false;
        }
        for (int i = 0; i < panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].
            Controls[MatchCnt.panelMatchRight.ToString()].Controls.Count; i++)
        {
            panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].
                Controls[MatchCnt.panelMatchRight.ToString()].Controls[i].Visible = false;
        }
        for (int i = 0; i < panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].
            Controls[MatchCnt.panelMatchingLines.ToString()].Controls.Count; i++)
        {
            panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].
                Controls[MatchCnt.panelMatchingLines.ToString()].Controls[i].Visible = false;
        }
    }
    for (int i = 0; i < panel3.Controls.Count; i++)
    {
        panel3.Controls[i].Visible = false;
    }
}

```



```

    }
    //panel3.Controls.Clear();
    //Отрисовка вопроса
    if (currentQuestion.Text.Count != 0)
    {
        if (!currentQuestion.graphicInQuestion)
        {
            ControlsForTextQuestion();
        }
        else
        {
            ControlsForImageQuestion();
            QuestionGroupBox.Controls[0].Text = currentQuestion.Text[0];
            (QuestionGroupBox.Controls[1] as PictureBox).Image = StringToImage(currentQuestion.Text[1]);
        }
    }
    else
    {
        EnableQuestion(true);
    }

    //Відображення питань
    if (currentQuestion.Answers.Count != 0)
    {
        if (currentQuestion.Answers[0] is AnswerText)
        {
            for (int i = 0; i < currentQuestion.Answers.Count; i++)
            {
                indexAnswer = i;
                EnableControlTextAnswer();
            }
        }
        if (currentQuestion.Answers[0] is AnswerMatch)
        {
            TableLayoutPanelForMatchAnswer();
            for (int i = 0; i < currentQuestion.Answers.Count; i++)
            {
                indexAnswer = i;
                EnableControlMatchAnswer();
            }
        }
    }
    else
    {
        variantToolStripMenuItem.Enabled = true;
        matchToolStripMenuItem.Enabled = true;
    }

    for (int i = 0; i < Manager.Instance.CurrentTest.variantCount; i++)
    {
        (panelCheckBoxVariant.Controls["VariantCheckbox_" + i] as CheckBox).Checked = currentQuestion.InVariant[i];
    }

    EnableToolStrip();

    timerUpdateElements.Start();
}

public void EnableToolStrip()
{
    EnableQuestion(currentQuestion.Text.Count == 0);
    if (currentQuestion.Answers.Count != 0)
    {
        if (currentQuestion.Answers[0] is AnswerText)
        {
            variantToolStripMenuItem.Enabled = true;
            matchToolStripMenuItem.Enabled = false;
        }
        else
        {
            matchToolStripMenuItem.Enabled = true;
            variantToolStripMenuItem.Enabled = false;
        }
    }
}

```

```

    }
    else
    {
        matchToolStripMenuItem.Enabled = true;
        variantToolStripMenuItem.Enabled = true;
    }
}

```

```

private void buttonUpdate_Click(object sender, EventArgs e)
{
    test.UpdateForm(Program.arg, indexQuestion);
}

```

//Видалення кнопки з питанням

```

private void slidedelToolStripMenuItem1_Click(object sender, EventArgs e)
{
    if (Manager.Instance.CurrentTest.Questions.Count > 1)
    {
        int num = Convert.ToInt32(indexQuestion);
        SlidesPanel.Controls.RemoveByKey(num.ToString());
        SlidesPanel.Controls.RemoveByKey("questionNumberLabel_" + num);
        Manager.Instance.CurrentTest.Questions.RemoveAt(indexQuestion);
        if (indexQuestion != 0)
            indexQuestion--;
        string[] splitStr;
        for (int i = 0; i < SlidesPanel.Controls.Count; i++)
        {
            splitStr = SlidesPanel.Controls[i].Name.Split('_');
            if (splitStr[0] != "questionNumberLabel")
            {
                int number = Convert.ToInt32(SlidesPanel.Controls[i].Name);
                if (number > num)
                {
                    SlidesPanel.Controls.SetChildIndex(SlidesPanel.Controls[number.ToString()], 0);
                    SlidesPanel.Controls[number.ToString()].Name = (number - 1).ToString();
                }
            }
            else
            {
                int number = Convert.ToInt32(splitStr[1]);
                if (number > num)
                {
                    SlidesPanel.Controls.SetChildIndex(SlidesPanel.Controls["questionNumberLabel_" + number], 0);
                    SlidesPanel.Controls["questionNumberLabel_" + number].Text = (number).ToString();
                    SlidesPanel.Controls["questionNumberLabel_" + number].Name = "questionNumberLabel_" + (number - 1);
                }
            }
        }
        for (int i = 0; i < SlidesPanel.Controls.Count / 2; i++)
        {
            SlidesPanel.Controls.SetChildIndex(SlidesPanel.Controls["questionNumberLabel_" + i], 0);
            SlidesPanel.Controls.SetChildIndex(SlidesPanel.Controls[i.ToString()], 0);
        }
        SlidesPanel.Controls[indexQuestion.ToString()].Focus();
        SlideGetFocus(SlidesPanel.Controls[indexQuestion.ToString()]);
    }
}

```

```

private void AddQuestionToolStripMenuItem_Click(object sender, EventArgs e)
{
    CreateSlideButton();
}

```

```

private void timerUpdateElements_Tick(object sender, EventArgs e)
{
    UpdateSlidePanelControls();
}

```

```

public void UpdateSlidePanelControls()
{
    //Кнопка-слайдом
}

```

```

Bitmap printscreen = new Bitmap(panel2.Width, panel2.Height);
panel2.DrawToBitmap(printscreen, new System.Drawing.Rectangle(0, 0, panel2.Width, panel2.Height));
for (int i = 0; i < SlidesPanel.Controls.Count; i++)
    if (SlidesPanel.Controls[i] is Button)
        if (SlidesPanel.Controls[i].Name == indexQuestion.ToString())
            (SlidesPanel.Controls[i] as Button).BackgroundImage = printscreen;
//test.UpdateForm(new List<string>(), indexQuestion);
}

private void button1_Click(object sender, EventArgs e)
{
    panel3.Controls[0].Visible = false;
}

private void contextMenuStripAnswers_Opening(object sender, System.ComponentModel.CancelEventArgs e)
{
}

private void tableLayoutPanel2_Paint(object sender, PaintEventArgs e)
{
}

private void numericUpDownVariantCount_ValueChanged(object sender, EventArgs e)
{
    int oldVarCount = Manager.Instance.CurrentTest.variantCount;
    Manager.Instance.CurrentTest.variantCount = Convert.ToInt32(numericUpDownVariantCount.Value);
    //panelCheckBoxVariant.Controls.Clear();
    for (int i = 0; i < Manager.Instance.CurrentTest.variantCount; i++)
    {
        if (panelCheckBoxVariant.Controls["VariantCheckbox_" + i] == null)
        {
            for (int j = 0; j < Manager.Instance.CurrentTest.Questions.Count; j++)
                Manager.Instance.CurrentTest.Questions[j].InVariant.Add(false);
            CheckBox variant = new CheckBox()
            {
                Dock = DockStyle.Left,
                Text = (i + 1).ToString(),
                Name = "VariantCheckbox_" + i,
                AutoSize = true
            };
            if (currentQuestion != null)
                variant.Checked = currentQuestion.InVariant[i];
            variant.CheckedChanged += Variant_CheckedChanged;
            panelCheckBoxVariant.Controls.Add(variant);
        }
    }
    for (int i = Manager.Instance.CurrentTest.variantCount; i < oldVarCount; i++)
    {
        panelCheckBoxVariant.Controls.RemoveByKey("VariantCheckbox_" + i);
        for (int j = 0; j < Manager.Instance.CurrentTest.Questions.Count; j++)
            Manager.Instance.CurrentTest.Questions[j].InVariant.RemoveAt(i);
    }
}

private void Variant_CheckedChanged(object sender, EventArgs e)
{
    string[] name = (sender as Control).Name.Split('_');
    currentQuestion.InVariant[Convert.ToInt32(name[1])] = (sender as CheckBox).Checked;
}

private void SaveFileButton_Click(object sender, EventArgs e)
{
    if (!saveAs)
    {
        for (int i = 0; i < Manager.Instance.CurrentTest.Questions.Count; i++)
        {
            Manager.Instance.CurrentTest.Questions[i].ImageSlide = SlidesPanel.Controls[i.ToString()].BackgroundImage;
        }
        ReadWriteTest.GetInstance().WriteTest(currentFileName);
    }
    else

```

```

        SaveAs();
    }

    private void buttonGoTest_Click(object sender, EventArgs e)
    {
        SetCurrentPanel(panelStartGoTest);
        for (int i = 0; i < Manager.Instance.CurrentTest.variantCount; i++)
        {
            comboBoxVariant.Items.Add(i + 1);
        }
        comboBoxVariant.Text = "1";
    }

    private void button1_Click_1(object sender, EventArgs e)
    {
        EditTest();
    }

    private void SavePassswordBut_Click(object sender, EventArgs e)
    {
        Manager.Instance.CurrentTest.password = textBoxChangePass.Text;
        MessageBox.Show("Успішно", "", MessageBoxButtons.OK, MessageBoxIcon.Information);
        SetCurrentPanel(MainWorkSpacePanel);
    }

    private void buttonStartGoTest_Click(object sender, EventArgs e)
    {
        //Program.arg.Add("C:\\Users\\Maxon\\Desktop\\тесты\\tests.tst");
        for (int i = 0; i < Program.arg.Count; i++)
        {
            if (Program.arg[i].Split('.')[Program.arg[i].Split('.').Length - 1] == "tst")
            {
                ReadWriteTest.GetInstance().ReadTest(Program.arg[i]);
                break;
            }
        }
        if (MessageBox.Show("Розпочати тестування", "Перевірка", MessageBoxButtons.OKCancel, MessageBoxIcon.Information) ==
        DialogResult.OK)
        {
            SetCurrentPanel(panelTestRun);
            Manager.Instance.SetQuestionsForTest(Convert.ToInt32(comboBoxVariant.Text));
            Manager.GoTest goTestQues = Manager.Instance.goTest;
            CreateButtonsReferenceToQuestions();
            Manager.Instance.goTest.indexQuestion = 0;
            ShowQuestion();
        }
    }

    public void CreateButtonsReferenceToQuestions()
    {
        for (int i = 0; i < Manager.Instance.goTest.questions.Count; i++)
        {
            Button nextPanel = new Button()
            {
                Name = i.ToString(),
                Text = (i + 1).ToString(),
                Size = new Size(32, 32),
                BackColor = Color.White,
                ForeColor = Color.Black,
                Location = new Point((i % 25) * 34 + 5, (int)(i / 25) * 34 + 5)
            };
            nextPanel.Font = new System.Drawing.Font("Times New Roman", 7);
            nextPanel.Click += NextPanel_Click;
            SlidesPanelInTest.Controls.Add(nextPanel);
        }
    }

    private void NextPanel_Click(object sender, EventArgs e)
    {
        Manager.Instance.goTest.indexQuestion = Convert.ToInt32((sender as Control).Name);
        ShowQuestion();
    }
}
/// <summary>

```

```

/// Контролі для графічного питання
/// </summary>
void GraphicControlQuestion(Question question)
{
    PictureBox questionPicture = new PictureBox()
    {
        Name = "questionPictureBox",
        Dock = DockStyle.Left,
        SizeMode = PictureBoxSizeMode.Zoom,
        Padding = new Padding(5),
        BackColor = Color.White,
        Size = new Size(300, 100),
        Image = StringToImage(question.Text[1])
    };
    panelQuestion.Controls.Add(questionPicture);
    panelQuestion.Controls.SetChildIndex(questionPicture, 0);
    questionPicture.Invalidate();
    Label questionTextBox = new Label()
    {
        Name = "labelQuestion",
        Dock = DockStyle.Top,
        Text = question.Text[0],
        ForeColor = Color.White,
        Font = new System.Drawing.Font("Times New Roman", 16),
        AutoSize = false
    };
    panelQuestion.Controls.Add(questionTextBox);
    panelQuestion.Controls.SetChildIndex(questionTextBox, 0);
}

/// <summary>
/// Контролі для текстового питання
/// </summary>
void TextControlQuestion(Question question)
{
    Label questionTextBox = new Label()
    {
        Name = "labelQuestion",
        Dock = DockStyle.Fill,
        Text = question.Text[0],
        ForeColor = Color.White,
        Font = new System.Drawing.Font("Times New Roman", 16),
        AutoSize = false,
    };
    panelQuestion.Controls.Add(questionTextBox);
    panelQuestion.Controls.SetChildIndex(questionTextBox, 0);
}

/// <summary>
/// Контролі для текстової відповіді
/// </summary>
/// <param name="question"></param>
void ControlForTextAnswer(Question question)
{
    for (int i = 0; i < question.Answers.Count; i++)
    {
        ///
        GroupBox groupBox = new GroupBox
        {
            Size = new Size(450, 44),
            Dock = DockStyle.Top,
            Padding = new Padding(10, 2, 2, 2),
            Name = "Answer_" + i,
            Font = DefaultFont,
        };
        ///
        CheckBox checkBox = new CheckBox
        {
            Size = new Size(25, 25),
            Dock = DockStyle.Right,
            Padding = new Padding(6, 0, 0, 5),
            Name = i.ToString(),
        };
    }
}

```

```

//Write userAnswer
checkBox.CheckedChanged += (sender, ev) =>
{
    (Manager.Instance.goTest.userQuestions[Manager.Instance.goTest.indexQuestion].
    Answers[Convert.ToInt32((sender as Control).Name)] as AnswerText).Right = (sender as CheckBox).Checked;
    for (int j = 0; j < panelAnswer.Controls.Count; j++)
    {
        if (panelAnswer.Controls["Answer_" + j].Controls[j.ToString()] is CheckBox)
        {
            if ((panelAnswer.Controls["Answer_" + j].Controls[j.ToString()] as CheckBox).Checked)
            {
                SlidesPanelInTest.Controls[Manager.Instance.goTest.indexQuestion].BackColor = Color.Gray;
                return;
            }
        }
    }
    SlidesPanelInTest.Controls[Manager.Instance.goTest.indexQuestion].BackColor = Color.White;
};
///
Label textBox = new Label
{
    Size = new Size(10, 20),
    Dock = DockStyle.Fill,
    Margin = new Padding(0),
    Name = "TextBox_" + indexAnswer,
    Text = (question.Answers[i] as AnswerText).Text,
    ForeColor = Color.White
};
groupBox.Controls.AddRange(new Control[] { checkBox, textBox });
groupBox.ContextMenuStrip = contextMenuStripAnswers;
panelAnswer.Controls.Add(groupBox);
panelAnswer.Controls.SetChildIndex(groupBox, 0);
}
}

/// <summary>
/// Контролі для відповіді-з'єднання
/// </summary>
void ControlForMatchAnswer(Question question)
{
    int yDelta = 0;
    for (int i = 0; i < question.Answers.Count; i++)
    {
        Label labelAnswerLeft = new Label()
        {
            Dock = DockStyle.Top,
            Text = (i + 1) + ". " + (question.Answers[i] as AnswerMatch).Text.Key,
            ForeColor = Color.White
        };
        Label labelAnswerRight = new Label()
        {
            Dock = DockStyle.Top,
            Text = Convert.ToChar(i + 1072) + ". " + (question.Answers[i] as AnswerMatch).Text.Value,
            ForeColor = Color.White,
            TextAlign = ContentAlignment.TopRight
        };
        panelAnswer.Controls.Add(labelAnswerLeft);
        panelAnswer.Controls.SetChildIndex(labelAnswerLeft, 0);

        panelAnswer.Controls.Add(labelAnswerRight);
        panelAnswer.Controls.SetChildIndex(labelAnswerRight, 0);

        yDelta = labelAnswerRight.Location.Y;
    }

    yDelta += 64;
    for (int i = 0; i < question.Answers.Count; i++)
    {
        Label label = new Label()
        {
            Text = (i + 1).ToString(),
            ForeColor = Color.White,
            Location = new Point(3, (i % 10) * 34 + 5 + yDelta),

```

```

        AutoSize = true
    };
    label.Font = new System.Drawing.Font("Times New Roman", 10);
    panelAnswer.Controls.Add(label);
    label = new Label()
    {
        Text = Convert.ToChar(i + 1072).ToString(),
        ForeColor = Color.White,
        Location = new Point(i * 34 + 38, yDelta - 16),
        AutoSize = true
    };
    label.Font = new System.Drawing.Font("Times New Roman", 10);
    panelAnswer.Controls.Add(label);
}

for (int i = 0; i < question.Answers.Count; i++)
{
    for (int j = 0; j < question.Answers.Count; j++)
    {
        Button nextPanel = new Button()
        {
            Name = i + "_" + j,
            //Text = i + "_" + j,
            Size = new Size(32, 32),
            BackColor = Color.White,
            ForeColor = Color.Black,
            Location = new Point(j * 34 + 32, (i % 10) * 34 + 5 + yDelta)
        };

        nextPanel.Click += (sender, ev) =>
        {
            Button button = (sender as Button);
            string[] buttonName = button.Name.Split('_');
            int x = Convert.ToInt32(buttonName[0]), y = Convert.ToInt32(buttonName[1]);
            (Manager.Instance.goTest.userQuestions[Manager.Instance.goTest.indexQuestion].Answers.
                Find(obj => (obj as AnswerMatch).Number == x) as AnswerMatch).RightVariant = y;
            for (int k = 0; k < question.Answers.Count; k++)
            {
                (panelAnswer.Controls.Find(x + "_" + k.ToString(), false)[0] as Button).BackColor = Color.White;
            }
            button.BackColor = Color.Gray;
            SlidesPanelInTest.Controls[Manager.Instance.goTest.indexQuestion].BackColor = Color.Gray;
        };
        nextPanel.Font = new System.Drawing.Font("Times New Roman", 7);
        panelAnswer.Controls.Add(nextPanel);
    }
}

private void comboBoxVariant_SelectedIndexChanged(object sender, EventArgs e)
{
}

private void SwitchQuestionButton_Click(object sender, EventArgs e)
{
    Manager.Instance.goTest.indexQuestion++;
    ShowQuestion();
}

/// <summary>
/// Create Controls for current question
/// </summary>
public void ShowQuestion()
{
    if (Manager.Instance.goTest.indexQuestion == Manager.Instance.goTest.questions.Count - 1)
        SwitchQuestionButton.Visible = false;
    else
        SwitchQuestionButton.Visible = true;

    Question question = Manager.Instance.goTest.questions[Manager.Instance.goTest.indexQuestion];
    //Створення елементів для питання
    panelQuestion.Controls.Clear();
}

```

```

        if (question.graphicInQuestion)
        {
            GraphicControlQuestion(question);
        }
        else
        {
            TextControlQuestion(question);
        }
        panelAnswer.Controls.Clear();
        if (question.Answers.Count != 0)
        {
            if (question.Answers[0] is AnswerText)
            {
                ControlForTextAnswer(question);
            }
            else
            {
                ControlForMatchAnswer(question);
            }
        }
    }
}

//Підрахунок результатів
private void buttonEndTest_Click(object sender, EventArgs e)
{
    Button curBut;
    int rightAns = 0, wrongAns = 0, ballCount = 0;
    int rightInOneAns = 0;
    int indexRightAns = 0;
    float percent;

    var document = new iTextSharp.text.Document();
    using (var writer = PdfWriter.GetInstance(document, new FileStream("result.pdf", FileMode.Create)))
    {
        document.Open();
        Paragraph p;

        iTextSharp.text.pdf.BaseFont bfR;
        bfR = iTextSharp.text.pdf.BaseFont.CreateFont("C:\\WINDOWS\\Fonts\\Arial.TTF",
            iTextSharp.text.pdf.BaseFont.IDENTITY_H,
            iTextSharp.text.pdf.BaseFont.EMBEDDED);
        iTextSharp.text.Font font = new iTextSharp.text.Font(bfR, 18, iTextSharp.text.Font.BOLD, BaseColor.BLACK);

        document.Add(new Paragraph("Протокол тестування.", font));
        document.Add(new Paragraph("Здобувач освіти: " + textBoxPIB.Text, font));
        font.Size = 14;
        p = new Paragraph();
        p.Alignment = Element.ALIGN_LEFT;
        p.TabSettings = new TabSettings(100f);
        p.Add Chunk.TABBING;
        p.Add(new Chunk("Правильний варіант", font));
        p.Add Chunk.TABBING;
        p.Add(new Chunk("Вибраний варіант", font));
        document.Add(p);
        font.SetStyle(iTextSharp.text.Font.NORMAL);
        for (int i = 0; i < Manager.Instance.goTest.userQuestions.Count; i++)
        {
            indexRightAns = 0;
            rightInOneAns = 0;
            //
            curBut = (SlidesPanelInTest.Controls[i.ToString()] as Button);
            curBut.Enabled = false;
            curBut.BackColor = Color.Green;
            //
            if (Manager.Instance.goTest.userQuestions[i].Answers.Count > 0)
            {
                //TextAnswer
                if (Manager.Instance.goTest.userQuestions[i].Answers[0] is AnswerText)
                {
                    //Find index of right answers
                    for (int j = 0; j < Manager.Instance.goTest.userQuestions[i].Answers.Count; j++)
                    {
                        //

```



```

        if ((Manager.Instance.goTest.questions[i].Answers[j] as AnswerText).Right)
        {
            indexRightAns = j;
            //
            p = new Paragraph();
            p.TabSettings = new TabSettings(150f);
            p.Add(Chunk.TABBING);
            p.Add(new Chunk((indexRightAns + 1).ToString(), font));
            ballCount++;
            //
        }
    }
    //if index of right answers == userAnswer index then ++ball
    for (int j = 0; j < Manager.Instance.goTest.userQuestions[i].Answers.Count; j++)
    {
        if ((Manager.Instance.goTest.userQuestions[i].Answers[j] as AnswerText).Right)
        {
            //
            p.TabSettings = new TabSettings(150f);
            p.Add(Chunk.TABBING);
            p.Add(new Chunk((j + 1).ToString(), font));
            document.Add(p);
            //
            if (j == indexRightAns)
                rightInOneAns++;
            else
            {
                wrongAns++;
                rightInOneAns = 0;
                break;
            }
        }
    }
    if (rightInOneAns == 0)
        curBut.BackColor = Color.Red;
    rightAns += rightInOneAns;
}
//MatchAnswer
else
{
    ballCount += Manager.Instance.goTest.questions[i].Answers.Count;
    //
    p = new Paragraph();
    p.TabSettings = new TabSettings(150f);
    p.Add(Chunk.TABBING);
    //
    //if index of right answers == userAnswer index then ++ball
    for (int j = 0; j < Manager.Instance.goTest.questions[i].Answers.Count; j++)
    {
        p.Add(new Chunk((j + 1).ToString() + ") - " + ((Manager.Instance.goTest.questions[i].Answers[j] as AnswerMatch).RightVariant
+ 1).ToString() + " ", font));
        //
        if ((Manager.Instance.goTest.userQuestions[i].Answers[j] as AnswerMatch).RightVariant ==
(Manager.Instance.goTest.questions[i].Answers[j] as AnswerMatch).RightVariant)
        {
            rightAns++;
            rightInOneAns++;
        }
        else
        {
            wrongAns++;
        }
    }
    for (int j = 0; j < Manager.Instance.goTest.userQuestions[i].Answers.Count; j++)
    {
        p.Add(new Chunk((j + 1).ToString() + ") - " + ((Manager.Instance.goTest.userQuestions[i].Answers[j] as
AnswerMatch).RightVariant + 1).ToString() + " ", font));
        //
    }
    //
    document.Add(p);
    //
    if (rightInOneAns == 0)

```

```

        {
            curBut.BackColor = Color.Red;
        }
        else if (rightInOneAns > 0 && rightInOneAns < Manager.Instance.goTest.userQuestions[i].Answers.Count)
        {
            curBut.BackColor = Color.Yellow;
        }
    }
}
//
percent = (float)rightAns / (ballCount) * 100f;
//
p = new Paragraph("\nВідсоток правильних відповідей: " + percent + " %", font);
p.Alignment = Element.ALIGN_RIGHT;
document.Add(p);
document.Close();
}

//
panelQuestion.Controls.Clear();
panelAnswer.Controls.Clear();
buttonEndTest.Visible = false;
SwitchQuestionButton.Visible = false;
Label labelRezult = new Label()
{
    Text = "Балів: " + rightAns + " з " + (ballCount) + " можливих",
    Dock = DockStyle.Top,
    AutoSize = true,
    Font = new System.Drawing.Font("Arial", 14),
    ForeColor = Color.White
};
panelQuestion.Controls.Add(labelRezult);

labelRezult.Text += Environment.NewLine + "Відсоток правильних відповідей: " + Math.Round(percent, 2) + " %";
}

private void toolStripButtonGoMenu_Click(object sender, EventArgs e)
{
    if (needSave)
    {
        DialogResult result = MessageBox.Show(
            "Зберегти зміни в " + currentFileName + "?",
            "ТЕСТ",
            MessageBoxButtons.YesNoCancel,
            MessageBoxIcon.Question,
            MessageBoxDefaultButton.Button1,
            MessageBoxOptions.DefaultDesktopOnly);
        if (result == DialogResult.Yes)
            Save();
        if (result != DialogResult.Cancel)
        {
            SetCurrentPanel(tableLayoutPanelStart);
            TopMost = true;
        }
        if (result == DialogResult.Cancel)
            TopMost = true;
    }
    else
    {
        SetCurrentPanel(tableLayoutPanelStart);
        TopMost = true;
    }
}

/// <summary>
/// LoadTestFromWord
/// </summary>
/// <param name="sender"></param>

```

```

/// <param name="e"></param>
private void toolStripButtonLoadTestFromWord_Click(object sender, EventArgs e)
{
    if (Manager.Instance.LoadQuestionsFromWord())
    {
        SlidesPanel.Controls.Clear();
        for (int j = 0; j < Manager.Instance.CurrentTest.variantCount; j++)
        {
            CheckBox variant = new CheckBox()
            {
                Dock = DockStyle.Left,
                Text = (j + 1).ToString(),
                Name = "VariantCheckbox_" + j,
                Checked = currentQuestion.InVariant[j],
                AutoSize = true
            };
            variant.CheckedChanged += Variant_CheckedChanged;
            panelCheckBoxVariant.Controls.Add(variant);
        }
        UIForTets();
    }
}

private void comboBoxSubject_SelectedIndexChanged(object sender, EventArgs e)
{
    Manager.Instance.CurrentTest.Questions[indexQuestion].subject = (Subject)Enum.Parse(typeof(Subject), (sender as ComboBox).Text);
}
}

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
using System.Drawing.Drawing2D;

namespace IsputCSharpWinFormsV2
{
    //Функції Вкладка ВСТАВИТИ

    public partial class MainForm
    {
        Graphics g;
        Graphics prev;

        // ТЕКСТОВА ВІДПОВІДЬ

        //додавання текстової відповіді
        private void варіантToolStripMenuItem_Click(object sender, EventArgs e)
        {
            indexAnswer = currentQuestion.Answers.Count;
            currentQuestion.Answers.Add(new AnswerText());
            (currentQuestion.Answers[indexAnswer] as AnswerText).Right = false;
            //ControlsForTextAnswer();
            if (matchToolStripMenuItem.Enabled)
            {
                matchToolStripMenuItem.Enabled = false;
                EnableControlTextAnswer();
            }

            public void EnableControlTextAnswer()
            {
                if (panel3.Controls["Answer_" + indexAnswer] == null)
                    ControlsForTextAnswer();
                panel3.Controls["Answer_" + indexAnswer].Visible = true;
                (panel3.Controls["Answer_" + indexAnswer].Controls["CheckBox_" + indexAnswer] as CheckBox).Checked =
                (currentQuestion.Answers[indexAnswer] as AnswerText).Right;
                panel3.Controls["Answer_" + indexAnswer].Controls["TextBox_" + indexAnswer].Text = (currentQuestion.Answers[indexAnswer] as
                AnswerText).Text;
                panel3.Controls.SetChildIndex(panel3.Controls["Answer_" + indexAnswer], 0);
            }
        }
    }
}

```

```

}

public void ControlsForTextAnswer()
{
    if (matchToolStripMenuItem.Enabled)
        matchToolStripMenuItem.Enabled = false;
    ///
    GroupBox groupBox = new GroupBox
    {
        Size = new Size(450, 44),
        Dock = DockStyle.Top,
        Padding = new Padding(10, 2, 2, 2),
        Name = "Answer_" + indexAnswer,
        Font = DefaultFont,
        Visible = false
    };
    ///
    CheckBox checkBox = new CheckBox
    {
        Size = new Size(25, 25),
        Dock = DockStyle.Right,
        Padding = new Padding(6, 0, 0, 5),
        Name = "CheckBox_" + indexAnswer,
        //Checked = (currentQuestion.Answers[indexAnswer] as AnswerText).Right
    };
    checkBox.CheckedChanged += (senderCheck, ev) =>
    {
        (currentQuestion.Answers[indexAnswer] as AnswerText).Right = (senderCheck as CheckBox).Checked;
    };
    ///
    TextBox textBox = new TextBox
    {
        Size = new Size(10, 20),
        Dock = DockStyle.Fill,
        Margin = new Padding(0),
        Name = "TextBox_" + indexAnswer,
        //Text = (currentQuestion.Answers[indexAnswer] as AnswerText).Text
    };
    textBox.TextChanged += (senderText, ev) =>
    {
        (currentQuestion.Answers[indexAnswer] as AnswerText).Text = (senderText as TextBox).Text;
    };
    groupBox.Controls.AddRange(new Control[] { checkBox, textBox });
    groupBox.ContextMenuStrip = contextMenuStripAnswers;
    panel3.Controls.Add(groupBox);
    panel3.Controls.SetChildIndex(groupBox, 0);
    for (int i = 0; i < panel3.Controls[groupBox.Name].Controls.Count; i++)
    {
        panel3.Controls[groupBox.Name].Controls[i].GotFocus += GroupBox_GotFocus;
    }
}

/// <summary>
/// Видалення текстового питання
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void видалитиToolStripMenuItem_Click(object sender, EventArgs e)
{
    int index = indexQuestion;
    if (currentQuestion.Answers.Count == 1)
    {
        matchToolStripMenuItem.Enabled = true;
    }
    currentQuestion.Answers.RemoveAt(indexAnswer);
    panel3.Controls.RemoveByKey("Answer_" + indexAnswer.ToString());
    int num = 0;
    string[] splitStr;
    for (int i = 0; i < panel3.Controls.Count; i++)
    {
        splitStr = panel3.Controls[i].Name.Split('_');
        if (splitStr[0] == "TextBox" || splitStr[0] == "CheckBox" || splitStr[0] == "Answer")
        {

```

```

        num = Convert.ToInt32(splitStr[1]);
        if (num > indexAnswer)
        {
            panel3.Controls[i].Controls["TextBox_" + num].Name = "TextBox_" + (num - 1);
            panel3.Controls[i].Controls["CheckBox_" + num].Name = "CheckBox_" + (num - 1);
            panel3.Controls[i].Name = "Answer_" + (num - 1);
        }
    }
}
indexQuestion = index;
SlideGetFocus(SlidesPanel.Controls[indexQuestion.ToString()]);
}
//

//ПИТАННЯ-З'ЄДНУВАЧ

//додавання питання з'єднувач
private void зєднанняToolStripMenuItem_Click(object sender, EventArgs e)
{
    indexAnswer = currentQuestion.Answers.Count;
    currentQuestion.Answers.Add(new AnswerMatch());
    if (variantToolStripMenuItem.Enabled)
    {
        variantToolStripMenuItem.Enabled = false;
    }
    EnableControlMatchAnswer();
}

public void EnableControlMatchAnswer()
{
    Control tablematchLayoutPanel = panel3.Controls[MatchCnt.matchLayoutPanel.ToString()];
    if (tablematchLayoutPanel.Visible == false)
    {
        tablematchLayoutPanel.Visible = true;
    }
    tablematchLayoutPanel.Controls[MatchCnt.panelMatchleft.ToString()].
        Controls[MatchCnt.groupBoxQuestionLeft + "_" + indexAnswer].Visible = true;

    tablematchLayoutPanel.Controls[MatchCnt.panelMatchleft.ToString()].
        Controls[MatchCnt.groupBoxQuestionLeft + "_" + indexAnswer].
        Controls["textLeft_" + indexAnswer].Text = (currentQuestion.Answers[indexAnswer] as AnswerMatch).Text.Key;

    tablematchLayoutPanel.Controls[MatchCnt.panelMatchRight.ToString()].
        Controls[MatchCnt.groupBoxQuestionRight + "_" + indexAnswer].
        Controls["textRight_" + indexAnswer].Text = (currentQuestion.Answers[indexAnswer] as AnswerMatch).Text.Value;

    tablematchLayoutPanel.Controls[MatchCnt.panelMatchRight.ToString()].
        Controls[MatchCnt.groupBoxQuestionRight + "_" + indexAnswer].Visible = true;
    tablematchLayoutPanel.Controls[MatchCnt.panelMatchingLines.ToString()].
        Controls[MatchCnt.RectMatchLeft + "_" + indexAnswer].Visible = true;
    tablematchLayoutPanel.Controls[MatchCnt.panelMatchingLines.ToString()].
        Controls[MatchCnt.RectMatchRight + "_" + indexAnswer].Visible = true;

    tablematchLayoutPanel.Controls[MatchCnt.panelMatchleft.ToString()].
        Controls.SetChildIndex(tablematchLayoutPanel.Controls[MatchCnt.panelMatchleft.ToString()].
            Controls[MatchCnt.groupBoxQuestionLeft + "_" + indexAnswer], 0);
    tablematchLayoutPanel.Controls[MatchCnt.panelMatchRight.ToString()].
        Controls.SetChildIndex(tablematchLayoutPanel.Controls[MatchCnt.panelMatchRight.ToString()].
            Controls[MatchCnt.groupBoxQuestionRight + "_" + indexAnswer], 0);
    (currentQuestion.Answers[indexAnswer] as AnswerMatch).Number = indexAnswer;
}

private void TableLayoutPanelForMatchAnswer()
{
    TableLayoutPanel matchLayoutPanel = new TableLayoutPanel()
    {
        Dock = DockStyle.Fill,
        RowCount = 1,
        ColumnCount = 3,
        Name = MatchCnt.matchLayoutPanel.ToString(),
        Visible = false
    };
    matchLayoutPanel.RowStyles.Add(new System.Windows.Forms.RowStyle(SizeType.Percent, 100f));

```

```

matchLayoutPanel.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(SizeType.Percent, 40f));
matchLayoutPanel.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(SizeType.Percent, 20f));
matchLayoutPanel.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(SizeType.Percent, 40f));
panel3.Controls.Add(matchLayoutPanel);
//Панель для відповідей праворуч
Panel panelRight = new Panel()
{
    Name = MatchCnt.panelMatchRight.ToString(),
    Dock = DockStyle.Fill,
    AutoScroll = true,
};
//Панель для відповідей ліворуч
Panel panelLeft = new Panel()
{
    Name = MatchCnt.panelMatchleft.ToString(),
    Dock = DockStyle.Fill,
    AutoScroll = true,
};
//Панель для з'єднання точок питань
PictureBox panelMatchingLines = new PictureBox()
{
    Name = MatchCnt.panelMatchingLines.ToString(),
    Dock = DockStyle.Fill,
};
panelMatchingLines.Paint += (senderS, ev) =>
{
    RedrawMatchingLines(true);
};
matchLayoutPanel.Controls.Add(panelRight, 2, 0);
matchLayoutPanel.Controls.Add(panelLeft, 0, 0);
matchLayoutPanel.Controls.Add(panelMatchingLines, 1, 0);
panelMatchingLines.MouseClick += (senderpan, ev) =>
{
    if (isStartMatching)
    {
        isStartMatching = false;
        mouseStart = new Point(mouseStart.X - 5, mouseStart.Y - 5);
        for (int j = 0; j < (senderpan as Control).Controls.Count; j++)
        {
            if ((senderpan as Control).Controls[j].Location == mouseStart)
            {
                ((senderpan as Control)).Controls[j].BackColor = Color.White;
            }
        }
    }
};
}

private void ControlsForMatchAnswer()
{
    Control.ControlCollection matchPanels = panel3.Controls["matchLayoutPanel"].Controls;
    //
    //Створення groupBox для питання справа
    GroupBox groupBoxQuestionLeft = new GroupBox()
    {
        Dock = DockStyle.Top,
        Text = /*MatchCnt.groupBoxQuestionLeft.ToString() + '_' + indexAnswer*/"",
        Size = new Size(300, 35),
        Padding = new Padding(3),
        Name = MatchCnt.groupBoxQuestionLeft.ToString() + '_' + indexAnswer,
        Visible = false
    };
    groupBoxQuestionLeft.ContextMenuStrip = contextMenuStripAnsMatch;
    TextBox textLeft = new TextBox()
    {
        Anchor = AnchorStyles.Left | AnchorStyles.Right | AnchorStyles.Top,
        Size = new Size(groupBoxQuestionLeft.Width - 27, 22),
        Location = new Point(5, 10),
        Name = "textLeft_" + indexAnswer
    };
    textLeft.TextChanged += (obj, EventArgs) =>
    {
        KeyValuePair<string, string> text = (currentQuestion.Answers[indexAnswer] as AnswerMatch).Text;
    }
}

```

```

        (currentQuestion.Answers[indexAnswer] as AnswerMatch).Text = new KeyValuePair<string, string>(textLeft.Text, text.Value);
    };
    textLeft.GotFocus += (obj, EventArgs) =>
    {
        indexAnswer = Convert.ToInt32((obj as Control).Name.Split('_')[1]);
    };
    Label labelLeft = new Label()
    {
        Dock = DockStyle.Right,
        Text = (indexAnswer + 1).ToString(),
        AutoSize = true,
        Name = "labelLeft_" + indexAnswer
    };
    //(currentQuestion.Answers[currentQuestion.Answers.Count - 1] as AnswerMatch).Number = Convert.ToInt32(labelLeft.Text);
    groupBoxQuestionLeft.Controls.AddRange(new Control[] { textLeft, labelLeft });
    matchPanels["panelMatchLeft"].Controls.Add(groupBoxQuestionLeft);
    matchPanels["panelMatchLeft"].Controls.SetChildIndex(groupBoxQuestionLeft, 0);
    //
    //Створення groupBox для питання зліва
    GroupBox groupBoxQuestionRight = new GroupBox()
    {
        Dock = DockStyle.Top,
        Text = "",
        Size = new Size(300, 35),
        Padding = new Padding(3),
        Name = "groupBoxQuestionRight_" + indexAnswer,
        Visible = false
    };
    groupBoxQuestionRight.ContextMenuStrip = contextMenuStripAnsMatch;
    TextBox textRight = new TextBox()
    {
        Anchor = AnchorStyles.Left | AnchorStyles.Right | AnchorStyles.Top,
        Size = new Size(groupBoxQuestionRight.Width - 27, 22),
        Location = new Point(25, 10),
        Name = "textRight_" + indexAnswer
    };
    textRight.TextChanged += (obj, EventArgs) =>
    {
        KeyValuePair<string, string> text = (currentQuestion.Answers[indexAnswer] as AnswerMatch).Text;
        (currentQuestion.Answers[indexAnswer] as AnswerMatch).Text = new KeyValuePair<string, string>(text.Key, textRight.Text);
    };
    textRight.GotFocus += (obj, EventArgs) =>
    {
        indexAnswer = Convert.ToInt32((obj as Control).Name.Split('_')[1]);
    };
    Label labelRight = new Label()
    {
        Dock = DockStyle.Left,
        Text = Convert.ToChar(matchPanels["panelMatchRight"].Controls.Count + 1072).ToString(),
        AutoSize = true,
        Name = "labelRight_" + indexAnswer
    };
    //(currentQuestion.Answers[currentQuestion.Answers.Count - 1] as AnswerMatch).Variant = labelLeft.Text[0];
    groupBoxQuestionRight.Controls.AddRange(new Control[] { textRight, labelRight });
    matchPanels["panelMatchRight"].Controls.Add(groupBoxQuestionRight);
    matchPanels["panelMatchRight"].Controls.SetChildIndex(groupBoxQuestionRight, 0);
    //
    //Створення панелі і кнопок для з'єднання відповідних питань
    int i = matchPanels["panelMatchingLines"].Controls.Count - 1;
    Panel matchLabelLeft = new Panel
    {
        Anchor = AnchorStyles.Left | AnchorStyles.Top,
        Name = "RectMatchLeft_" + indexAnswer,
        Text = "",
        AutoSize = true,
        Size = new Size(10, 10),
        BackColor = Color.White,
        Visible = false
    };
    matchLabelLeft.MouseClick += Label_Click;
    //matchLabelLeft.ContextMenuStrip = contextMenuStripMatchPanels;
    if (i == -1)

```

```

        matchLabelLeft.Location = new Point(5, 15);
    else
        matchLabelLeft.Location = new Point(5, matchPanels["panelMatchingLines"].Controls[i].Location.Y + 35);
    matchPanels["panelMatchingLines"].Controls.Add(matchLabelLeft);
    Panel matchLabelRight = new Panel
    {
        Anchor = AnchorStyles.Right | AnchorStyles.Top,
        Name = "RectMatchRight_" + indexAnswer,
        Text = "",
        AutoSize = true,
        Size = new Size(10, 10),
        BackColor = Color.White,
        Visible = false
    };
    matchLabelRight.MouseClick += Label_Click;
    //matchLabelRight.ContextMenuStrip = contextMenuStripMatchPanels;
    if (i == -1)
        matchLabelRight.Location = new Point(matchPanels["panelMatchingLines"].Width - 15, 15);
    else
        matchLabelRight.Location = new Point(matchPanels["panelMatchingLines"].Width - 15,
matchPanels["panelMatchingLines"].Controls[i].Location.Y + 35);
    matchPanels["panelMatchingLines"].Controls.Add(matchLabelRight);
    g = matchPanels["panelMatchingLines"].CreateGraphics();
    matchPanels["panelMatchingLines"].MouseMove += (senderML, ev) =>
    {
        if (isStartMatching)
        {
            RedrawMatchingLines();
            g.DrawLine(new Pen(Brushes.White, 2), mouseStart, ev.Location);
        }
    };
}

//Видалення питання з'єднувач
private void deletetoolStripMenuItemMatchAns_Click(object sender, EventArgs e)
{
    int index = indexQuestion;
    if (currentQuestion.Answers.Count == 1)
    {
        variantToolStripMenuItem.Enabled = true;
        panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].Visible = false;
    }
    Control.ControlCollection matchlayout = panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].Controls;
    int number = 0;
    bool needDecRV = false;
    for (int i = 0; i < currentQuestion.Answers.Count; i++)
    {
        needDecRV = false;
        if ((currentQuestion.Answers[i] as AnswerMatch).Number - 1 == indexAnswer)
        {
            number = (currentQuestion.Answers[i] as AnswerMatch).RightVariant;
            matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls[MatchCnt.RectMatchLeft + "_" + i].BackColor = Color.White;
            if (number != -1)
                matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls[MatchCnt.RectMatchRight + "_" + number].BackColor =
Color.White;
        }

        if ((currentQuestion.Answers[i] as AnswerMatch).RightVariant == indexAnswer)
        {
            number = (currentQuestion.Answers[i] as AnswerMatch).RightVariant;
            matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls[MatchCnt.RectMatchLeft + "_" + i].BackColor = Color.White;
            matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls[MatchCnt.RectMatchRight + "_" + number].BackColor = Color.White;
            needDecRV = true;
        }

        if ((currentQuestion.Answers[i] as AnswerMatch).RightVariant > indexAnswer)
        {
            (currentQuestion.Answers[i] as AnswerMatch).RightVariant -= 1;
        }
        if ((currentQuestion.Answers[i] as AnswerMatch).Number - 1 > indexAnswer)
        {
            (currentQuestion.Answers[i] as AnswerMatch).Number -= 1;
        }
    }
}

```



```

        if (needDecRV)
        {
            (currentQuestion.Answers[i] as AnswerMatch).RightVariant = -1;
        }
    }

    currentQuestion.Answers.RemoveAt(indexAnswer);

    matchlayout[MatchCnt.panelMatchleft.ToString()].Controls.
        RemoveByKey(MatchCnt.groupBoxQuestionLeft.ToString() + "_" + indexAnswer);
    matchlayout[MatchCnt.panelMatchRight.ToString()].Controls.
        RemoveByKey(MatchCnt.groupBoxQuestionRight.ToString() + "_" + indexAnswer);
    matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls.
        RemoveByKey(MatchCnt.RectMatchLeft.ToString() + "_" + indexAnswer);
    matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls.
        RemoveByKey(MatchCnt.RectMatchRight.ToString() + "_" + indexAnswer);

    int num = 0;
    string[] splitStr;
    for (int i = 0; i < matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls.Count; i++)
    {
        splitStr = matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls[i].Name.Split('_');
        num = Convert.ToInt32(splitStr[1]);
        if (num > indexAnswer)
        {
            Point loc = matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls[i].Location;

            for (int j = 0; j < matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls.Count; j++)
            {
                if (matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls[j].Location.Y == loc.Y)
                {
                    matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls[j].Location = new Point(loc.X, loc.Y - 35);
                    break;
                }
            }
            matchlayout[MatchCnt.panelMatchingLines.ToString()].Controls[i].Location = new Point(loc.X, loc.Y - 35);
        }
    }
    string controlName = "";
    for (int j = 0; j < 3; j++)
    {
        if (j == 0)
            controlName = MatchCnt.panelMatchleft.ToString();
        if (j == 1)
            controlName = MatchCnt.panelMatchRight.ToString();
        if (j == 2)
            controlName = MatchCnt.panelMatchingLines.ToString();
        for (int i = 0; i < matchlayout[controlName].Controls.Count; i++)
        {
            splitStr = matchlayout[controlName].Controls[i].Name.Split('_');
            num = Convert.ToInt32(splitStr[1]);
            if (num > indexAnswer)
            {
                if (j == 0)
                {
                    matchlayout[controlName].Controls[i].Controls[MatchCnt.labelLeft + "_" + num].Text = (num).ToString();
                    matchlayout[controlName].Controls[i].Controls[MatchCnt.textLeft + "_" + num].Name =
                        MatchCnt.textLeft.ToString() + "_" + (num - 1);
                    matchlayout[controlName].Controls[i].Controls[MatchCnt.labelLeft + "_" + num].Name =
                        MatchCnt.labelLeft.ToString() + "_" + (num - 1);
                    matchlayout[controlName].Controls[i].Name = MatchCnt.groupBoxQuestionLeft + "_" + (num - 1);
                }
                if (j == 2)
                {
                    if (splitStr[0] == MatchCnt.RectMatchLeft.ToString())
                        matchlayout[controlName].Controls[i].Name = MatchCnt.RectMatchLeft + "_" + (num - 1);
                    else
                        matchlayout[controlName].Controls[i].Name = MatchCnt.RectMatchRight + "_" + (num - 1);
                }
                if (j == 1)
                {
                    matchlayout[controlName].Controls[i].Controls[MatchCnt.labelRight + "_" + num].Text = Convert.ToChar(num - 1 +
1072).ToString();

```

```

        matchlayout[controlName].Controls[i].Controls[MatchCnt.textRight + "_" + num].Name =
            MatchCnt.textRight.ToString() + "_" + (num - 1);
        matchlayout[controlName].Controls[i].Controls[MatchCnt.labelRight + "_" + num].Name =
            MatchCnt.labelRight.ToString() + "_" + (num - 1);
        matchlayout[controlName].Controls[i].Name = MatchCnt.groupBoxQuestionRight + "_" + (num - 1);
    }
}
}
}

RedrawMatchingLines();
indexQuestion = index;
SlideGetFocus(SlidesPanel.Controls[indexQuestion.ToString()]);
}

//

private void RedrawMatchingLines(bool needCreate = true)
{
    try
    {
        if (currentQuestion.Answers.Count != 0)
        {
            if (currentQuestion.Answers[0] is AnswerMatch)
            {
                if (g != null)
                {
                    if (needCreate)
                    {
                        try
                        {
                            g = panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].
                                Controls[MatchCnt.panelMatchingLines.ToString()].CreateGraphics();
                        }
                        catch (Exception e)
                        {
                        }
                    }
                }
                g.Clear(Color.FromArgb(24, 73, 107));
                for (int j = 0; j < currentQuestion.Answers.Count; j++)
                {
                    int right = (currentQuestion.Answers[j] as AnswerMatch).RightVariant;
                    if (right != -1)
                    {
                        Point startP = panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].
                            Controls[MatchCnt.panelMatchingLines.ToString()].
                                Controls[MatchCnt.RectMatchLeft.ToString() + '_' + (j).ToString()].Location;
                        Point endP = panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].
                            Controls[MatchCnt.panelMatchingLines.ToString()].
                                Controls[MatchCnt.RectMatchRight.ToString() + '_' + (right).ToString()].Location;
                        startP = new Point(startP.X + 5, startP.Y + 5);
                        endP = new Point(endP.X + 5, endP.Y + 5);
                        g.DrawLine(new Pen(Brushes.White, 2), startP, endP);
                    }
                }
            }
            for (int i = 0; i < panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].
                Controls[MatchCnt.panelMatchingLines.ToString()].
                    Controls.Count; i++)
            {
                panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].
                    Controls[MatchCnt.panelMatchingLines.ToString()].
                        Controls[i].BackColor = Color.White;
            }
            for (int i = 0; i < currentQuestion.Answers.Count; i++)
            {
                if ((currentQuestion.Answers[i] as AnswerMatch).RightVariant != -1)
                {
                    panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].
                        Controls[MatchCnt.panelMatchingLines.ToString()].
                            Controls[MatchCnt.RectMatchLeft + "_" + i].BackColor = Color.Green;
                    panel3.Controls[MatchCnt.matchLayoutPanel.ToString()].

```

```

Controls[MatchCnt.panelMatchingLines.ToString()].
Controls[MatchCnt.RectMatchRight + "_" + (currentQuestion.Answers[i] as AnswerMatch).RightVariant].BackColor =
Color.Green;
    }
    }
    }
    }
    catch (Exception e)
    {
    }

}

private void GroupBox_GotFocus(object sender, EventArgs e)
{
    string name = (sender as Control).Name;
    string[] namesplit = name.Split('_');
    indexAnswer = Convert.ToInt32(namesplit[1]);
}

//Для з'єднання відповідей
private void Label_Click(object sender, MouseEventArgs e)
{
    Control senderObj = sender as Control;
    string[] name = senderObj.Name.Split('_');
    if (senderObj.BackColor != Color.Green)
    {
        if (isStartMatching)
        {
            if (senderObj.Location.X + 5 != mouseStart.X)
            {
                mouseEnd = new Point(senderObj.Location.X + 5, senderObj.Location.Y + 5);
                RedrawMatchingLines();
                g.DrawLine(new Pen(Brushes.White, 2), mouseStart, mouseEnd);
                isStartMatching = false;
                senderObj.BackColor = Color.Green;
                if (name[0] == MatchCnt.RectMatchLeft.ToString())
                    numberMatch = Convert.ToInt32(name[1]);
                if (name[0] == MatchCnt.RectMatchRight.ToString())
                    variantMatch = Convert.ToInt32(name[1]);
                (currentQuestion.Answers[numberMatch] as AnswerMatch).RightVariant = variantMatch;
            }
        }
        else
        {
            mouseStart = new Point(senderObj.Location.X + 5, senderObj.Location.Y + 5);
            senderObj.BackColor = Color.Green;
            isStartMatching = true;
            if (name[0] == MatchCnt.RectMatchLeft.ToString())
                numberMatch = Convert.ToInt32(name[1]);
            if (name[0] == MatchCnt.RectMatchRight.ToString())
                variantMatch = Convert.ToInt32(name[1]);
        }
    }
    else if (!isStartMatching)
    {
        senderObj.BackColor = Color.White;
        isStartMatching = true;
        Control.ControlCollection coll = panel3.Controls[MatchCnt.matchLayoutPanel.ToString()];
        Controls[MatchCnt.panelMatchingLines.ToString()].Controls;
        if (name[0] == MatchCnt.RectMatchLeft.ToString())
        {
            variantMatch = (currentQuestion.Answers[Convert.ToInt32(name[1])] as AnswerMatch).RightVariant;
            (currentQuestion.Answers[Convert.ToInt32(name[1])] as AnswerMatch).RightVariant = -1;
            mouseStart = coll[MatchCnt.RectMatchRight.ToString() + "_" + variantMatch].Location;
            mouseStart = new Point(mouseStart.X + 5, mouseStart.Y + 5);
        }
        if (name[0] == MatchCnt.RectMatchRight.ToString())
        {
            int num = Convert.ToInt32(name[1]);
            numberMatch = (currentQuestion.Answers.FindIndex(answer => (answer as AnswerMatch).RightVariant == num));
            (currentQuestion.Answers[numberMatch] as AnswerMatch).RightVariant = -1;
            mouseStart = coll[MatchCnt.RectMatchLeft.ToString() + "_" + numberMatch].Location;
            mouseStart = new Point(mouseStart.X + 5, mouseStart.Y + 5);
        }
    }
}

```

```

    }
    RedrawMatchingLines();
}
}

//ПИТАНИЯ
//Додати питання текст
private void textToolStripMenuItem_Click(object sender, EventArgs e)
{
    currentQuestion.Text.Add("");
    ControlsForTextQuestion();
}

public void ControlsForTextQuestion()
{
    QuestionGroupBox.ContextMenuStrip = contextMenuStripQuesDel;
    imageToolStripMenuItem.Enabled = false;
    textToolStripMenuItem.Enabled = false;
    currentQuestion.graphicInQuestion = false;
    TextBox QuestionText = new TextBox()
    {
        Name = "questionTextBox",
        Dock = DockStyle.Fill,
        Multiline = true,
        Text = currentQuestion.Text[0]
    };
    QuestionGroupBox.Controls.Add(QuestionText);
    //QuestionText.Focus();
    QuestionText.TextChanged += (senderQuestion, ev) =>
    {
        currentQuestion.Text[0] = (senderQuestion as TextBox).Text;
    };
}

//Додати питання картинку
private void imageToolStripMenuItem_Click(object sender, EventArgs e)
{
    currentQuestion.Text.Add("");
    currentQuestion.Text.Add("");
    ControlsForImageQuestion();
    ReadImage();
}

/// <summary>
/// Зчитування картинки для питання
/// </summary>
public void ReadImage()
{
    using (OpenFileDialog open_dialog = new OpenFileDialog())
    {
        open_dialog.Filter = "Image Files (*.BMP;*.JPG;*.GIF;*.PNG)|*.BMP;*.JPG;*.GIF;*.PNG|All files (*.*)|*.*"; //формат завантаженого
        if (open_dialog.ShowDialog() == DialogResult.OK) //если в окне была нажата кнопка "OK"
        {
            try
            {
                Image image = Image.FromFile(open_dialog.FileName);
                (QuestionGroupBox.Controls[1] as PictureBox).Image = image;
                currentQuestion.Text[1] = ImageToString(image);
            }
            catch
            {
                DialogResult result = MessageBox.Show("Невозможно открыть выбранный файл",
                    "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}

/// <summary>
/// Створення UIлем для граф питання

```

```

/// </summary>
public void ControlsForImageQuestion()
{
    QuestionGroupBox.ContextMenuStrip = contextMenuStripQuesDel;
    textToolStripMenuItem.Enabled = false;
    imageToolStripMenuItem.Enabled = false;
    currentQuestion.graphicInQuestion = true;
    PictureBox questionPicture = new PictureBox()
    {
        Name = "questionPictureBox",
        //Image = image,
        Dock = DockStyle.Left,
        SizeMode = PictureBoxSizeMode.Zoom,
        Padding = new Padding(5),
        BackColor = Color.White
    };
    questionPicture.MouseClick += (sen, ev1) =>
    {
        ReadImage();
    };
    QuestionGroupBox.Controls.Add(questionPicture);
    QuestionGroupBox.Controls.SetChildIndex(questionPicture, 0);
    questionPicture.Invalidate();
    TextBox questionTextBox = new TextBox()
    {
        Name = "questionTextBox",
        Multiline = true,
        Dock = DockStyle.Top,
    };
    QuestionGroupBox.Controls.Add(questionTextBox);
    QuestionGroupBox.Controls.SetChildIndex(questionTextBox, 0);
    //questionTextBox.Focus();

    questionTextBox.TextChanged += (senderQuestion, ev) =>
    {
        currentQuestion.Text[0] = (senderQuestion as TextBox).Text;
    };
}

//Конвертувати string to Image
private Image StringToImage(string imageString)
{
    if (imageString != "")
    {
        byte[] imageBytes;
        // Convert Base64 String to byte[]
        imageBytes = Convert.FromBase64String(imageString);
        MemoryStream ms = new MemoryStream(imageBytes, 0,
            imageBytes.Length);
        // Convert byte[] to Image
        ms.Write(imageBytes, 0, imageBytes.Length);
        return Image.FromStream(ms, true);
    }
    else
        return null;
}

//Конвертировать Image to string
private string ImageToString(Image image)
{
    byte[] imageBytes;
    string imageString;
    using (MemoryStream mss = new MemoryStream())
    {
        // Convert Image to byte[]
        image.Save(mss, System.Drawing.Imaging.ImageFormat.Png);
        imageBytes = mss.ToArray();
        // Convert byte[] to Base64 String
        imageString = Convert.ToBase64String(imageBytes);
    }
    return imageString;
}

```

```

        //Дозвіл чи заборона додавання питання
        private void EnableQuestion(bool enable)
        {
            textToolStripMenuItem.Enabled = enable;
            imageToolStripMenuItem.Enabled = enable;
        }

        private void CheckBtnEnable()
        {
        }

        private void delQuestionToolStripMenuItem_Click(object sender, EventArgs e)
        {
            currentQuestion.Text.Clear();
            QuestionGroupBox.Controls.Clear();
            textToolStripMenuItem.Enabled = true;
            imageToolStripMenuItem.Enabled = true;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
using System.Drawing.Drawing2D;
using System.Threading;
using System.Runtime.InteropServices;
using System.Diagnostics;

namespace IsputCSharpWinFormsV2
{
    public partial class MainForm : Form
    {
        public string currentFileName = "";
        public bool needSave;
        public bool saveAs;

        private void OpenFileButton_Click(object sender, EventArgs e)
        {
            // Displays an OpenFileDialog so the user can select a Cursor.
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.Filter = "Cursor Files|*.tst";
            openFileDialog.Title = "Select a Cursor File";
            // Show the Dialog.
            // If the user clicked OK in the dialog and
            // a .CUR file was selected, open it.

            if (openFileDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                ProcessStartInfo startInfo = new ProcessStartInfo();
                startInfo.FileName = "IsputCSharpWinFormsV2.exe";
                startInfo.Arguments = openFileDialog.FileName;
                Process.Start(startInfo);
            }
        }

        public void UIForTets()
        {
            for (int i = 0; i < Manager.Instance.CurrentTest.Questions.Count; i++)
            {
                CreateUIForSlide(i + 1);
                SlidesPanel.Controls[i.ToString()].BackgroundImage = Manager.Instance.CurrentTest.Questions[i].ImageSlide;
            }
            SlideGetFocus(this.SlidesPanel.Controls["0"]);
        }

        //Збергти тест
        private void toolStripButtonSave_Click(object sender, EventArgs e)
        {

```

```

        Save();
        needSave = false;
        saveAs = false;
    }

    public void Save()
    {
        if (!saveAs)
        {
            for (int i = 0; i < Manager.Instance.CurrentTest.Questions.Count; i++)
            {
                Manager.Instance.CurrentTest.Questions[i].ImageSlide = SlidesPanel.Controls[i.ToString()].BackgroundImage;
            }
            ReadWriteTest.GetInstance().WriteTest(currentFileName);
        }
        else
            SaveAs();
    }

    private void SaveFileAsButton_Click(object sender, EventArgs e)
    {
        SaveAs();
    }

    private void SaveAs()
    {
        for (int i = 0; i < Manager.Instance.CurrentTest.Questions.Count; i++)
        {
            Manager.Instance.CurrentTest.Questions[i].ImageSlide = SlidesPanel.Controls[i.ToString()].BackgroundImage;
        }
        // Displays an OpenFileDialog so the user can select a Cursor.
        SaveFileDialog openFileDialog = new SaveFileDialog();
        openFileDialog.Filter = "Cursor Files|*.tst";
        openFileDialog.Title = "Select a Cursor File";

        // Show the Dialog.
        // If the user clicked OK in the dialog and
        // a .CUR file was selected, open it.
        if (openFileDialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        {
            string f = openFileDialog.FileName;
            ReadWriteTest.GetInstance().WriteTest(f);
            currentFileName = f;
            needSave = false;
            saveAs = false;
        }
    }

    private void CreateFileButton_Click(object sender, EventArgs e)
    {
        ProcessStartInfo startInfo = new ProcessStartInfo();
        startInfo.FileName = "IsputCSharpWinFormsV2.exe";
        startInfo.Arguments = "";
        Process.Start(startInfo);
    }
}

using System;
using System.Collections.Generic;
using System.Text;

namespace IsputCSharpWinFormsV2
{
    //Имена контролов
    enum VariantCnt
    {
        textLeft, labelLeft, textRight, labelRight
    }

    enum MatchCnt
    {

```

```

        matchLayoutPanel, panelMatchRight, panelMatchleft, panelMatchingLines, RectMatchLeft, RectMatchRight,
        groupBoxQuestionLeft, groupBoxQuestionRight, textLeft, labelLeft, textRight, labelRight
    }

    public enum Subject
    {
        Відсутній, Матем, Алгоритм
    }
}

using System;
using System.Collections.Generic;
using System.Text;
using System.Windows.Forms;
//using Microsoft.Office.Interop.Word;

namespace IspuCSharpWinFormsV2
{
    class Manager
    {
        private static Manager InstanceVar;

        public Test CurrentTest { get; set; }

        public GoTest goTest;
        public Dictionary<int, int> rating;

        Manager()
        {
            CurrentTest = new Test();
            goTest = new GoTest();
            rating = new Dictionary<int, int>();
            //
            rating.Add(1, 0);
            rating.Add(2, 5);
            rating.Add(3, 10);
            rating.Add(4, 20);
            rating.Add(5, 40);
            rating.Add(6, 50);
            rating.Add(7, 60);
            rating.Add(8, 70);
            rating.Add(9, 75);
            rating.Add(10, 80);
            rating.Add(11, 85);
            rating.Add(12, 100);
            //
        }

        public void SetQuestionsForTest(int variant)
        {
            goTest.variant = variant;
            for (int i = 0; i < CurrentTest.Questions.Count; i++)
            {
                if (CurrentTest.Questions[i].InVariant[goTest.variant - 1])
                {
                    goTest.questions.Add(new Question()
                    {
                        Text = CurrentTest.Questions[i].Text,
                        Answers = new List<Answer>(),
                        ImageSlide = CurrentTest.Questions[i].ImageSlide,
                        graphicInQuestion = CurrentTest.Questions[i].graphicInQuestion,
                        subject = CurrentTest.Questions[i].subject
                    });
                    for (int j = 0; j < CurrentTest.Questions[i].Answers.Count; j++)
                    {
                        if (CurrentTest.Questions[i].Answers.Count > 0)
                        {
                            if (CurrentTest.Questions[i].Answers[0] is AnswerText)
                            {
                                goTest.questions[goTest.questions.Count - 1].Answers.Add(new AnswerText()
                                {
                                    Right = (CurrentTest.Questions[i].Answers[j] as AnswerText).Right,

```



```

                Text = (CurrentTest.Questions[i].Answers[j] as AnswerText).Text
            });
        }
        else
        {
            goTest.questions[goTest.questions.Count - 1].Answers.Add(new AnswerMatch()
            {
                Number = (CurrentTest.Questions[i].Answers[j] as AnswerMatch).Number,
                RightVariant = (CurrentTest.Questions[i].Answers[j] as AnswerMatch).RightVariant,
                Text = (CurrentTest.Questions[i].Answers[j] as AnswerMatch).Text
            });
        }
    }
}
}
for (int i = 0; i < goTest.questions.Count; i++)
{
    goTest.userQuestions.Add(new Question()
    {
        Text = goTest.questions[i].Text,
        Answers = new List<Answer>(),
        subject = goTest.questions[i].subject
    });
    for (int j = 0; j < goTest.questions[i].Answers.Count; j++)
    {
        if (goTest.questions[i].Answers.Count > 0)
        {
            if (goTest.questions[i].Answers[0] is AnswerText)
            {
                goTest.userQuestions[goTest.userQuestions.Count - 1].Answers.Add(new AnswerText()
                {
                    Right = false,
                    Text = (goTest.questions[i].Answers[j] as AnswerText).Text
                });
            }
            else
            {
                goTest.userQuestions[goTest.userQuestions.Count - 1].Answers.Add(new AnswerMatch()
                {
                    Number = (goTest.questions[i].Answers[j] as AnswerMatch).Number,
                    RightVariant = -1,
                    Text = (goTest.questions[i].Answers[j] as AnswerMatch).Text
                });
            }
        }
    }
}
}
}

public static Manager Instance
{
    get
    {
        if (InstanceVar == null)
            InstanceVar = new Manager();
        return InstanceVar;
    }
}

public class GoTest
{
    public int variant;
    public List<Question> questions;
    public List<Question> userQuestions;
    public int indexQuestion;

    public GoTest()
    {
        questions = new List<Question>();
        userQuestions = new List<Question>();
        variant = 0;
        indexQuestion = 0;
    }
}

```

```

    }
}
/// <summary>
/// Load Questions From Word
/// </summary>
public bool LoadQuestionsFromWord()
{
    StringBuilder text = new StringBuilder();
    Microsoft.Office.Interop.Word.Application word = new Microsoft.Office.Interop.Word.Application();
    object miss = System.Reflection.Missing.Value;
    // Displays an OpenFileDialog so the user can select a Cursor.
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "(*.doc)|*.doc|(*.docx)|*.docx";
    openFileDialog.Title = "Select a Cursor File";
    // Show the Dialog.
    // If the user clicked OK in the dialog and
    // a .CUR file was selected, open it.

    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        object path = openFileDialog.FileName; // @"C:\Users\Maxon\Desktop\тесты\тест за вариантами — копия.doc";
        object readOnly = true;
        Microsoft.Office.Interop.Word.Document docs = word.Documents.Open(ref path, ref miss, ref readOnly, ref miss, ref miss, ref miss, ref miss, ref miss, ref miss, ref miss, ref miss, ref miss, ref miss, ref miss, ref miss, ref miss);
        string[] docsStrings = docs.Content.Text.Split('\r');
        docs.Close();
        //docsStrings split to Test
        int currentVariant = 0, indexQuestion = 0, k;
        string nextLine = "";
        CurrentTest.Questions.Clear();
        for (int i = 0; i < docsStrings.Length; i++)
        {
            nextLine = docsStrings[i];
            if (nextLine.Split(' ')[0].ToLower() == "вариант" && int.TryParse(nextLine.Split(' ')[1], out currentVariant))
            {
                if (currentVariant > CurrentTest.variantCount)
                {
                    CurrentTest.variantCount++;
                    for (int j = 0; j < CurrentTest.Questions.Count; j++)
                    {
                        CurrentTest.Questions[j].InVariant.Add(false);
                    }
                }
            }
            else if (int.TryParse(nextLine.Split('.')[0], out k))
            {
                CurrentTest.Questions.Add(new Question());
                indexQuestion = CurrentTest.Questions.Count - 1;
                CurrentTest.Questions[indexQuestion].Text.Add(nextLine);
                CurrentTest.Questions[indexQuestion].InVariant[currentVariant - 1] = true;
            }
            else if (nextLine != "")
            {
                CurrentTest.Questions[indexQuestion].Answers.Add(new AnswerText()
                {
                    Text = nextLine
                });
            }
        }
        return true;
    }
    else
    {
        return false;
    }
}

using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.Serialization;

```

```

using System.IO;
using System.Xml.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
using System.Runtime.Serialization.Formatters;

namespace IsputCSharpWinFormsV2
{
    class ReadWriteTest
    {
        private static ReadWriteTest Instance;
        public void ReadTest(string filePath)
        {
            ///через XML
            /*
            // передаем в конструктор тип класса
            XmlSerializer formatter = new XmlSerializer(typeof(Test));
            //
            StreamReader streamReader = new StreamReader("myFile.tst");
            Manager.GetInstance().CurrentTest = (Test)formatter.Deserialize(streamReader);
            streamReader.Dispose();
            streamReader.Close();*/

            ///через бинарный форматер
            BinaryFormatter binaryFormatter = new BinaryFormatter();
            Manager.Instance.CurrentTest.Questions.Clear();
            using (FileStream fs = new FileStream(filePath, FileMode.Open))
            {
                Manager.Instance.CurrentTest = (Test)binaryFormatter.Deserialize(fs);
            }

            if (Manager.Instance.CurrentTest.password == null)
                Manager.Instance.CurrentTest.password = "1111";
        }
        public void WriteTest(string fileName)
        {
            ///Через XML
            /*
            // передаем в конструктор тип класса
            XmlSerializer formatter = new XmlSerializer(typeof(Test));
            StreamWriter streamWriter = new StreamWriter("myFile.tst", false);
            formatter.Serialize(streamWriter, Manager.GetInstance().CurrentTest);
            streamWriter.Dispose();
            streamWriter.Close();
            */
            /// через бинарный форматер
            // создаем объект BinaryFormatter
            BinaryFormatter formatter = new BinaryFormatter();
            // получаем поток, куда будем записывать сериализованный объект

            using (FileStream fs = new FileStream(fileName, FileMode.OpenOrCreate))
            {
                formatter.Serialize(fs, Manager.Instance.CurrentTest);
            }
        }
        public static ReadWriteTest GetInstance()
        {
            if (Instance == null)
                Instance = new ReadWriteTest();
            return Instance;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Runtime.InteropServices;

namespace IsputCSharpWinFormsV2
{

```

```

static class Program
{
    /// <summary>
    /// Главная точка входа для приложения.
    /// </summary>
    static public List<string> arg;
    [STAThread]
    static void Main(string[] args)
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        arg = new List<string>();
        for (int i = 0; i < args.Length; i++)
        {
            arg.Add(args[i]);
        }
        MainForm main = new MainForm();
        Application.Run(main);
    }
}
using System;
using System.Collections.Generic;
using System.Text;

namespace IsputCSharpWinFormsV2
{
    [Serializable]
    public class Test
    {
        public List<Question> Questions { get; set; }
        public int variantCount;
        public string password;
        public Test()
        {
            Questions = new List<Question>();
            variantCount = 1;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
using System.Drawing.Drawing2D;
using System.Threading;
using System.Runtime.InteropServices;

//Kiyashko
namespace IsputCSharpWinFormsV2
{
    [Serializable]
    public class Question
    {
        public List<string> Text { get; set; }
        public List<Answer> Answers { get; set; }
        public bool graphicInQuestion { get; set; }
        public Image ImageSlide { get; set; }

        public List<bool> InVariant { get; set; }
        public Subject subject;

        public Question()
        {
            Text = new List<string>();
            Answers = new List<Answer>();
            InVariant = new List<bool>();
            for (int i = 0; i < Manager.Instance.CurrentTest.variantCount; i++)
            {

```

```

        InVariant.Add(false);
    }
    subject = Subject.Відсутній;
}
public int AnswersCount
{
    get { return Answers.Count; }
}
}

[Serializable]
public abstract class Answer {
    public abstract string Show();
}
[Serializable]
public class AnswerText:Answer
{
    public bool Right { get; set; }
    public string Text { get; set; }
    public AnswerText() { }

    public override string Show()
    {
        return "Right " + Right + " Text " + Text + Environment.NewLine;
    }
}

[Serializable]
public class AnswerMatch:Answer
{
    public int Number { get; set; }
    //public char Variant { get; set; }
    public KeyValuePair<string, string> Text { get; set; }
    public int RightVariant { get; set; }
    public AnswerMatch()
    {
        Text = new KeyValuePair<string, string>();
        RightVariant = -1;
    }
    public override string Show()
    {
        return "Number " + Number + " Text " + Text.Key + "," + Text.Value +
            " RightVariant " + RightVariant + Environment.NewLine;
    }
}
}

using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Runtime.InteropServices;

namespace IsputCSharpWinFormsV2
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        static public List<string> arg;
        [STAThread]
        static void Main(string[] args)
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            arg = new List<string>();
            for (int i = 0; i < args.Length; i++)
            {
                arg.Add(args[i]);
            }
            MainForm main = new MainForm();
            Application.Run(main);
        }
    }
}

```

```

    }
}
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace IsputCSharpWinFormsV2
{
    public partial class MainTestForm : Form
    {
        //Для перемещения окна
        private bool isDragging = false;
        private Point lastCursor;
        private Point lastForm;
        //-----
        //Для изменения размера окна
        private const int cGrip = 32; // Grip size
        private const int cCaption = 32; // Caption bar height;
        //-----
        public MainTestForm()
        {
            InitializeComponent();
            this.FormBorderStyle = FormBorderStyle.None;
            this.DoubleBuffered = true;
            this.SetStyle(ControlStyles.ResizeRedraw, true);

            for (int i = 1; i <= 5; i++)
                Temp(i);
        }
        protected override void OnPaint(PaintEventArgs e)
        {
            Rectangle rc = new Rectangle(this.ClientSize.Width - cGrip, this.ClientSize.Height - cGrip, cGrip, cGrip);
            ControlPaint.DrawSizeGrip(e.Graphics, this.BackColor, rc);
            rc = new Rectangle(0, 0, this.ClientSize.Width, cCaption);
            e.Graphics.FillRectangle(Brushes.DarkBlue, rc);
        }
        protected override void WndProc(ref Message m)
        {
            if (m.Msg == 0x84)
            { // Trap WM_NCHITTEST
                Point pos = new Point(m.LParam.ToInt32());
                pos = this.PointToClient(pos);
                if (pos.Y < cCaption)
                {
                    m.Result = (IntPtr)2; // HTCAPTION
                    return;
                }
                if (pos.X >= this.ClientSize.Width - cGrip && pos.Y >= this.ClientSize.Height - cGrip)
                {
                    m.Result = (IntPtr)17; // HTBOTTOMRIGHT
                    return;
                }
            }
            base.WndProc(ref m);
        }
        private void Header_MouseDown(object sender, MouseEventArgs e)
        {
            isDragging = true;
            lastCursor = Cursor.Position;
            lastForm = Location;
        }
        private void Header_MouseMove(object sender, MouseEventArgs e)
        {
            if (isDragging)
            {
                this.Location =
                    Point.Add(lastForm, new Size(Point.Subtract(Cursor.Position, new Size(lastCursor))));
            }
        }
    }
}

```

```

    }
}
private void Header_MouseUp(object sender, MouseEventArgs e)
{
    isDragging = false;
}
private void CloseWindowButton_Click(object sender, EventArgs e)
{
    Close();
}
private void MinimizeWindowButton_Click(object sender, EventArgs e)
{
    WindowState = FormWindowState.Minimized;
}
private void RestoreWindowButton_Click(object sender, EventArgs e)
{
    if (WindowState == FormWindowState.Maximized)
    {
        WindowState = FormWindowState.Normal;
        (sender as Button).BackgroundImage = IsputCSharpWinFormsV2.Properties.Resources.Maximize_Window_32px;
    }
    else if (WindowState == FormWindowState.Normal)
    {
        WindowState = FormWindowState.Maximized;
        (sender as Button).BackgroundImage = IsputCSharpWinFormsV2.Properties.Resources.Restore_Window_32px;
    }
}

private void Temp(int i)
{
    //Manager.GetInstance().AddQuestion();
    //int num = Manager.GetInstance().Questions.Count;
    //num -= 1;
    Button button = new Button
    {
        //Name = num.ToString(),
        //BackgroundImage = printPanel,
        Text = i.ToString(),

        BackgroundImageLayout = ImageLayout.Stretch,
        Padding = new Padding(0),
        Margin = new Padding(0),
        Dock = DockStyle.Left,
        FlatStyle = FlatStyle.Flat
    };
    button.ForeColor = Color.White;
    button.FlatAppearance.BorderSize = 1;
    button.FlatAppearance.BorderColor = Color.FromArgb(17, 57, 83);

    /*pictureBox.MouseEnter += delegate (object sender2, EventArgs e2)
    {
        (sender2 as Button).FlatAppearance.BorderColor = Color.FromArgb(60, 155, 200);
    };*/
    button.MouseEnter += (object sender2, EventArgs e2) =>
    {
        //if (Convert.ToInt32((sender2 as Button).Name) != Manager.GetInstance().ActiveQuestion)
        (sender2 as Button).FlatAppearance.BorderColor = Color.FromArgb(60, 155, 200);
    };
    button.MouseLeave += delegate (object sender2, EventArgs e2)
    {
        //if (Convert.ToInt32((sender2 as Button).Name) != Manager.GetInstance().ActiveQuestion)
        (sender2 as Button).FlatAppearance.BorderColor = Color.FromArgb(17, 57, 83);
    };
    button.MouseClick += delegate (object sender2, MouseEventArgs e2)
    {
        //for (int i = 0; i < SlidesPanel.Controls.Count; i++)
        // (SlidesPanel.Controls[i] as Button).FlatAppearance.BorderColor = Color.FromArgb(17, 57, 83);

        (sender2 as Button).FlatAppearance.BorderColor = Color.White;
        //Manager.GetInstance().ActiveQuestion = Convert.ToInt32(((sender2 as Button).Name));
    };
}

```

```

List<Control> slides = new List<Control>();
foreach (Control nextButton in SlidesPanelInTest.Controls)
{
    slides.Add(nextButton);
}

SlidesPanelInTest.Controls.Clear();
SlidesPanelInTest.Controls.Add(button);

button.Size = new Size(button.Size.Height, button.Size.Height);
button.Font = new Font("Microsoft Sans Serif", button.Size.Height / 2);
foreach (Control item in slides)
{
    SlidesPanelInTest.Controls.Add(item);
}
}
private void SlidesPanelInTest_SizeChanged(object sender, EventArgs e)
{
    //ControlCollection controlCollection = panel1.Controls;
    foreach (Control c in SlidesPanelInTest.Controls)
    {
        //c.Size = new Size(c.Size.Width, (int)(c.Size.Width * 0.75f));
        c.Size = new Size(c.Size.Height, c.Size.Height);
        c.Font = new Font("Microsoft Sans Serif", c.Size.Height/2);
    }
}
}
}

using System;
using System.Collections.Generic;
using System.Text;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace IsputCSharpWinFormsV2
{
    public enum TabName { FileTab, MainTab, InsertTab }
    public partial class MainForm
    {
        private ToolStrip activeToolStrip;
        private Button activeTabButton;
        void SetTab(TabName tabName)
        {
            if ((activeTabButton != null) && (tabName.ToString() + "Button" == activeTabButton.Name.ToString()))
            {
                activeToolStrip.Visible = false;
                activeTabButton.BackColor = Color.FromArgb(17, 57, 83);
                activeTabButton.ForeColor = Color.White;
                activeTabButton = null;
                activeToolStrip = null;
                return;
            }
            if (activeToolStrip != null)
                activeToolStrip.Visible = false;
            if (activeTabButton != null)
            {
                activeTabButton.BackColor = Color.FromArgb(17, 57, 83);
                activeTabButton.ForeColor = Color.White;
            }
            switch (tabName)
            {
                case TabName.FileTab:

                    break;
                case TabName.MainTab:
                    activeTabButton = MainTabButton;

```



```

        activeToolStrip = ToolStripMain;
        break;
    case TabName.InsertTab:
        activeTabButton = InsertTabButton;
        activeToolStrip = ToolStripInsert;
        break;
    }
    activeTabButton.BackColor = Color.FromArgb(241, 241, 241);
    activeTabButton.ForeColor = Color.FromArgb(17, 57, 83);
    activeToolStrip.Visible = true;
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
using System.Drawing.Drawing2D;

```

```

namespace IsputCSharpWinFormsV2
{
    public partial class MainForm
    {
        /// <summary>
        private Point _start;
        private Point _end;
        private bool _drawMode;
        private readonly List<GraphicsPath> _arrows;

        //Для перемещения окна
        private bool isDragging = false;
        private Point lastCursor;
        private Point lastForm;
        //-----

        //Для изменения размера окна
        private const int cGrip = 32;    // Grip size
        private const int cCaption = 32; // Caption bar height;
        //-----

        private void MainForm_Paint(object sender, PaintEventArgs e)
        {
            using (var pen = new Pen(Color.Red) { DashStyle = DashStyle.Dash })
            {
                e.Graphics.DrawPath(pen, GetArrow(_start, _end, 25, 20));
            }
            foreach (var arrow in _arrows)
            {
                e.Graphics.DrawPath(Pens.Blue, arrow);
            }
        }

        private void MainForm_MouseDown(object sender, MouseEventArgs e)
        {
            _start = e.Location;
            _drawMode = true;
        }

        private void MainForm_MouseUp(object sender, MouseEventArgs e)
        {
            _end = e.Location;
            _drawMode = false;
            _arrows.Add(GetArrow(_start, _end, 25, 20));
            Invalidate();
        }

        private void MainForm_MouseMove(object sender, MouseEventArgs e)
        {
            if (!_drawMode) return;
            _end = e.Location;
            Invalidate();
        }
    }
}

```

```

    }

    /// <summary>
    /// Рисувание отрезка со стрелкой посередине
    /// </summary>
    /// <param name="start">Начало отрезка</param>
    /// <param name="end">Конец отрезка</param>
    /// <param name="h">Раствор стрелки</param>
    /// <param name="l">Длина стрелки</param>
    private GraphicsPath GetArrow(Point start, Point end, float h, float l)
    {
        var gp = new GraphicsPath();
        //Вектора
        var v = new PointF(end.X - start.X, end.Y - start.Y);
        //Длина
        float len = (float)Math.Sqrt(v.X * v.X + v.Y * v.Y);
        //Нормированный вектор
        var norm = new PointF(v.X / len, v.Y / len);
        //середина отрезка.
        var mid = new PointF((end.X + start.X) / 2f, (end.Y + start.Y) / 2f);
        //Отступаем на длину стрелки
        var mid1 = new PointF(mid.X - l * norm.X, mid.Y - l * norm.Y);
        //Нормали для раствора стрелки
        var n1 = new PointF(-norm.Y * h / 2 + mid1.X, norm.X * h / 2 + mid1.Y);
        var n2 = new PointF(norm.Y * h / 2 + mid1.X, -norm.X * h / 2 + mid1.Y);
        gp.AddLine(start, end);
        gp.StartFigure();
        gp.AddLine(n1, mid);
        gp.AddLine(mid, n2);
        return gp;
    }

    protected override void OnPaint(PaintEventArgs e)
    {
        Rectangle rc = new Rectangle(this.ClientSize.Width - cGrip, this.ClientSize.Height - cGrip, cGrip, cGrip);
        ControlPaint.DrawSizeGrip(e.Graphics, this.BackColor, rc);
        rc = new Rectangle(0, 0, this.ClientSize.Width, cCaption);
        e.Graphics.FillRectangle(Brushes.DarkBlue, rc);
    }

    protected override void WndProc(ref Message m)
    {
        if (m.Msg == 0x84)
        {
            // Trap WM_NCHITTEST
            Point pos = new Point(m.LParam.ToInt32());
            pos = this.PointToClient(pos);
            if (pos.Y < cCaption)
            {
                {
                    m.Result = (IntPtr)2; // HTCAPTION
                    return;
                }
            }
            if (pos.X >= this.ClientSize.Width - cGrip && pos.Y >= this.ClientSize.Height - cGrip)
            {
                {
                    m.Result = (IntPtr)17; // HTBOTTOMRIGHT
                    return;
                }
            }
        }
        RedrawMatchingLines(true);
        base.WndProc(ref m);
    }
    private void Header_MouseDown(object sender, MouseEventArgs e)
    {
        isDragging = true;
        lastCursor = Cursor.Position;
        lastForm = Location;
    }
    private void Header_MouseMove(object sender, MouseEventArgs e)
    {
        if (isDragging)
        {
            this.Location =
                Point.Add(lastForm, new Size(Point.Subtract(Cursor.Position, new Size(lastCursor))));
        }
    }

```

```

    }
}
private void Header_MouseUp(object sender, MouseEventArgs e)
{
    isDragging = false;
}
private void CloseWindowButton_Click(object sender, EventArgs e)
{
    if (needSave)
    {
        DialogResult result = MessageBox.Show(
            "Збергти зміни в " + currentFileName + "?",
            "ТЕСТ",
            MessageBoxButtons.YesNoCancel,
            MessageBoxIcon.Question,
            MessageBoxDefaultButton.Button1,
            MessageBoxOptions.DefaultDesktopOnly);
        if (result == DialogResult.Yes)
            Save();
        if (result != DialogResult.Cancel)
            Close();
        if (result == DialogResult.Cancel)
            TopMost = true;
    }
    else
        Close();
}
private void MinimizeWindowButton_Click(object sender, EventArgs e)
{
    WindowState = FormWindowState.Minimized;
}
private void RestoreWindowButton_Click(object sender, EventArgs e)
{
    if (WindowState == FormWindowState.Maximized)
    {
        WindowState = FormWindowState.Normal;
        (sender as Button).BackgroundImage = IsputCSharpWinFormsV2.Properties.Resources.Maximize_Window_32px;
    }
    else if (WindowState == FormWindowState.Normal)
    {
        WindowState = FormWindowState.Maximized;
        (sender as Button).BackgroundImage = IsputCSharpWinFormsV2.Properties.Resources.Restore_Window_32px;
    }
}
private void TabButton_Click(object sender, EventArgs e)
{
    switch ((sender as Button).Name)
    {
        case "FileTabButton":
            SwitchMainPanel();
            break;
        case "MainTabButton":
            SetTab(TabName.MainTab);
            break;
        case "InsertTabButton":
            SetTab(TabName.InsertTab);
            break;
    }
}

private void SwitchMainPanel()
{
    MainWorkSpacePanel.Visible = !MainWorkSpacePanel.Visible;
    LeftMenuAndInfPanel.Visible = !LeftMenuAndInfPanel.Visible;
}
private void ButtonBack_Click(object sender, EventArgs e)
{
    SwitchMainPanel();
}

private void panel1_SizeChanged(object sender, EventArgs e)
{
    //ControlCollection controlCollection = panel1.Controls;

```

```

        foreach (Control c in SlidesPanel.Controls)
        {
            if (c is Button)
                c.Size = new Size(c.Size.Width, (int)(c.Size.Width * 0.75f));
        }

    }

    private void Header_DoubleClick(object sender, EventArgs e)
    {
        if (WindowState == FormWindowState.Maximized)
        {
            WindowState = FormWindowState.Normal;
            ButtonMaximizeWindow.BackgroundImage = IsputCSharpWinFormsV2.Properties.Resources.Maximize_Window_32px;
        }
        else if (WindowState == FormWindowState.Normal)
        {
            WindowState = FormWindowState.Maximized;
            ButtonMaximizeWindow.BackgroundImage = IsputCSharpWinFormsV2.Properties.Resources.Restore_Window_32px;
        }
    }

    //private void ReDraw()
    //{
    //    RedrawMatchingLines();
    //    thread = new System.Threading.Thread(this.ReDraw);
    //    thread.Start();
    //    System.Threading.Thread.Sleep(0);
    //}

    }
}

```