ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ БІЗНЕСУ ТА СУЧАСНИХ ТЕХНОЛОГІЙ

ФОРМА НАВЧАННЯ ДЕННА
КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту

Завідувач кафедри _____О.О. Ємець

«_____»_____202_ р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему

Створення тренажера з теми «Коди із виправленням помилок» дистанційного навчального курсу «Теорія інформації і кодування»

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Купченко Олексій Володимирович
_____«___»_____2021р.

Науковий керівник к.ф.-м.н., доц., Парфьонова Тетяна Олександрівна
_____ «___»_____2021р.

ПОЛТАВА 2021 р.

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

ЗАТВЕРДЖУЮ

Завідувач кафедри_____О.О. Ємець
« 8 **»** вересня 2020р.

Завдання та календарний графік
виконання дипломної роботи

Студент спеціальності 122 «Комп'ютерні науки»
Прізвище, ім'я, по батькові Купченко Олексій Володимирович
1. Тема **«Створення тренажера з теми «Коди із виправленням помилок» дистанційного навчального курсу «Теорія інформації і кодування»»**
затверджена наказом ректора № 121-Н від « 1 » вересня 2020 р.
Термін подання студентом бакалаврської роботи « 20 » травня 2021 р.

2. Вихідні дані до дипломної роботи: публікації з теми навчальні тренажери в дистанційних курсах з комп'ютерних наук, матеріали дистанційного курсу «Теорія інформації і кодування», а також умови задач, з яких буде складатися тренажер.

Задача: Побудова циклічного коду. Закодувати двійковим циклічним кодом, що виправляє однократні помилки, комбінацію 10100 двійкового простого коду.

Крім розв'язування заданих задач, тренажер має реалізувати повторення теоретичного матеріалу по темі, зокрема таких понять, як код, кодова комбінація, коректувальний код, помилка, кодування, декодування, синдром, кодова відстань.

3. Зміст пояснювальної записки

Вступ
1. ПОСТАНОВКА ЗАДАЧІ
1.1. Формулювання прикладу, на якому розробляється тренажер
1.2. Вимоги до тренажера

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд робіт, де розглянуте аналогічне до теми роботи завдання.

2.2. Позитивні аспекти оглянутих робіт.

2.3. Вади розробок з оглянутих робіт.

2.4. Необхідність та актуальність теми роботи.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Алгоритмізація задачі за темою роботи

3.2. Розробка блок-схеми, яка підлягає програмуванню.

3.3. Обґрунтування вибору програмних засобів

4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис процесу програмної реалізації.

4.2. Опис програми.

4.3. Тестування програми

4.4. Необхідна користувачу програми інструкція.

Висновки

4. Перелік графічного матеріалу: 5-6 аркушів блок-схем, інші необхідні ілюстрації.

5. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1. Постанова задачі	Парфьонова Т.О., доц.	8.09.20	8.09.20
2. Інформаційний огляд	Парфьонова Т.О., доц.	8.09.20	8.09.20
3. Теоретична частина	Парфьонова Т.О., доц.	8.09.20	8.09.20
4. Практична реалізація	Парфьонова Т.О., доц.	8.09.20	8.09.20

6. Календарний графік виконання бакалаврської роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ	10.05.21	
2. Вивчення методичних рекомендацій та стандартів та звіт керівнику	15.09.20	

Зміст роботи	Термін виконання	Фактичне виконання
3. Постановка задачі	1.10.20	
4. Інформаційний огляд джерел бібліотек та інтернету	2.11.20	
5. Теоретична частина	1.02.21	
6. Практична частина	17.05.21	
7. Закінчення оформлення	21.05.21	
8. Доповідь студента на кафедрі	28.05.21	
9. Доробка (за необхідністю), рецензування	14.06.21	

Дата видачі завдання «8» вересня 2020 р.

Студент Купченко Олексій Володимирович

Науковий керівник _____ к.ф.-м.н., доц., Парфьонова Тетяна Олександрівна
(підпис)

Результати захисту бакалаврської роботи
Дипломна робота оцінена на
_____
(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 2021 р.

Секретар ЕК_____ _____
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Записка: 97 с., 44 рис., 7 таблиці, 1 додаток (на 44 сторінках), 16 джерел.

Предмет розробки – програма-тренажер для побудови кодів із виправленням помилок.

Мета роботи – розробка графічного програмного застосунку-тренажеру для роботи з алгоритмами кодування із виправленням помилок.

Методи, які були використані для розв'язування задачі – Методи побудови кодів из виправлення помилок. Робоча програма розроблена в середовищі Lazarus.

Розроблено графічну програму-тренажер для алгоритмів кодування із виправленням помилок.

Здійснена програмна реалізація за допомогою мови програмування Object Pascal бібліотеки графічних елементів інтерфейсу користувача LCL та в інтегрованому середовищі розробки Lazarus.

Ключові слова: КОД, КОДОВА КОМБІНАЦІЯ, КОРЕКТУВАЛЬНИЙ КОД, ПОМИЛКА, КОДУВАННЯ, ДЕКОДУВАННЯ, СИНДРОМ, КОДОВА ВІДСТАНЬ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, термінів
ІС	Інформаційна система
ПЗ	Програмне забезпечення
ОР	Object Pascal
IDE	Integrated Development Environment

ВСТУП

З появою технічних засобів для передачі та збереження інформації повстала проблема з підтримкою її цілісності під час її передачі, обробки та зберіганні. Для цього були створені спеціальні алгоритми для знаходження та виправлення помилок, так звані коректувальні коди. На сьогоднішній день, швидкий розвиток інформаційних технологій та мережі Інтернет посприяли стрімкому збільшенню кількості інформації, що в свою чергу призвело до нагальної потреби у висококваліфікованих фахівцях з «Теорії інформації та кодування».

У той самий час старі методи навчання не дають можливості однаково швидко та якісно навчати велику кількість висококваліфікованих фахівців з «Теорії інформації та кодування».

Об'єктом дослідження даної роботи є програмний тренажер з теми «Коди із виправленням помилок». Актуальність даної роботи, полягає в прискоренні і спрощенні навчання більшої кількості студентів з предмету «Теорія інформації та кодування», а також можливості автоматизувати процес навчання, а саме: вирішення задач з предмету та їх пояснення.

Метою даної роботи є розробка програмного забезпечення у вигляді тренажеру для навчання студентів навчального курсу «Теорія інформації та кодування» по темі «Коди із виправленням помилок», а саме проектування і розробка графічного інтерфейсу програми та розробка алгоритму вирішення задачі по темі «Коди із виправленням помилок».

З практичної точки зору створення тренажеру у вигляді програмного забезпечення з графічним інтерфейсом дає переваги у якості, швидкості та зручності одночасного навчання великої кількості студентів.

1 ПОСТАНОВКА ЗАДАЧІ

1.1 Формулювання прикладу, на якому розробляється тренажер

В якості прикладу, на основі якого створюється тренажер, було обрано наступну задачу з теми «Коди із виправленням помилок»: закодувати двійковим циклічним кодом, що виправляє однократні помилки, комбінацію 10100 двійкового простого коду.

Початок алгоритму.

Крок 1. Виводиться текст завдання: «Встановіть відповідність» та надаються задані позначеннями k, r, та n та варіанти відповідей, які подані у вигляді списку, що випадає:

– довжина кодової комбінації;

– кількість інформаційних елементів у комбінації;

– кількість перевірних елементів у комбінації.

Якщо було обрано одну або декілька неправильних відповідей, то з'являється повідомлення про помилку: «Правильні відповіді наступні: k – кількість інформаційних елементів у комбінації, r – кількість перевірних елементів у комбінації, n – довжина кодової комбінації.» [1].

Переходимо на крок 2.

Крок 2. Виводиться текст завдання: «Виберіть умову, якою пов'язані k, r, та n» та надається список варіантів відповідей, представлений у таблиці 1.1.

Таблиця 1.1 – Варіанти відповідей на Кроці 2

Правильна відповідь	Відповіді
+	$n = k + r$
–	$n = k - r$
–	$n = 2 * k + r$
–	$k + r = 2 * n$

Якщо було обрано неправильну відповідь, то виводиться повідомлення: «Довжина кодової комбінації – це сума кількості інформаційних елементів і перевірних, таким чином $n = k + r$.».

Переходимо на крок 3.

Крок 3. Виводиться текст завдання: «Введіть правильне значення у комірку» та надається комірка для вводу відповіді. Введене значення порівнюється з правильною відповіддю «5».

Якщо введене значення не співпадає з правильною відповіддю, то з'являється повідомлення: «Кількість інформаційних елементів у комбінації дорівнює кількості елементів у заданій комбінації, що кодується».

Надається можливість повторного введення. Якщо користувач знову помиляється, то виводиться повідомлення про помилку, і клітина отримує автоматичне правильне значення «5».

Переходимо на крок 4.

Крок 4. Виводиться текст завдання: «Введіть правильне значення у комірку» та надається комірка для вводу відповіді. Введене значення порівнюється з правильною відповіддю «4».

Якщо введене значення не співпадає з правильною відповіддю, то з'являється повідомлення: «Кількість перевірних елементів r у комбінації циклічного коду, визначається згідно умови $2^r - 1 \geq n$ або $2^r \geq k + r + 1$. У даному випадку $2^r - 1 \geq r + 6$» [2].

Надається можливість повторного введення, якщо знову допущено помилку, то виводиться повідомлення: «Найменше значення r, що задовольняє умові: $2^r - 1 \geq r + 6$ дорівнює 4. Таким чином $2^4 - 1 \geq 4 + 6$; $15 > 10$. Тому $r = 4$.» і клітина отримає правильне значення.

Переходимо на крок 5.

Крок 5. Надається таблиця, або її графічне зображення, твірних поліномів, представлена в таблиці 1.2 [3].

Таблиця 1.2 – Твірні поліноми

r	Твірний поліном $P(x)$	Двійковий запис полінома
1	$x+1$	11
2	x^2+x+1	111
3	x^3+x+1	1011
4	x^3+x^2+1	1101
4	x^4+x+1	10011
4	x^4+x^3+1	11001
4	$x^4+x^3+x^2+x+1$	11111
5	x^5+x^2+1	100101
5	x^5+x^3+1	101001
5	$x^5+x^3+x^2+x+1$	101111
5	$x^5+x^4+x^2+x+1$	110111
6	$x^6+x^5+x^4+1$	1110001
8	$x^8+x^7+x^6+x^5+x^2+x+1$	111100111
9	$x^9+x^5+x^3+1$	1000101001
15	$x^{15}+x^{14}+x^{13}+x^{12}+x^4+x^3+x^2+x+1$	1111000000011111
16	$x^{16}+x^{12}+x^5+1$	10001000000100001

Також виводиться текст завдання: «Який з твірних поліномів *P(x)* можна використати для подальшого кодування? Виберіть правильні варіанти.» та набір правильних відповідей представлений у таблиці 1.3.

Таблиця 1.3 – Варіанти відповідей на Кроці 5

Правильні варіанти	Варіанти відповідей
–	$P(x)=x^2+x+1$
+	$P(x)=x^4+x+1$
–	$P(x)=x^5+x^3+1$
–	$P(x)=x^3+x+1$
+	$P(x)=x^4+x^3+1$

Якщо було зроблено помилковий вибір, то виводиться повідомлення: «Порядок полінома $P(x)$ (старший степінь) визначається кількістю перевірних елементів» та надається можливість повторного вибору.

Користувач здійснює повторний вибір. В разі помилки, виводиться повідомлення про помилку: «Вибір неправильний» і автоматично відображаються правильні відповіді. Переходимо на крок 6.

Крок 6. Виводиться текст завдання: «Вибрати поліном $Q(x)$, що відповідає комбінації 10100, що кодується.» та надається список відопідей, представлений у таблиці 1.4.

Таблиця 1.4 – Варіанти відповідей на Кроці 6

Правильна відповідь	Відповіді
–	$Q(x) = x^5 + x^3$
–	$Q(x) = x^3 + x + 1$
+	$Q(x) = x^4 + x^2$
–	$Q(x) = x^4 + x^2 + x$

Якщо було зроблено помилковий вибір, то виводиться повідомлення: «Поліном $Q(x)$, що відповідає комбінації 10100: $Q(x) = x^4 + x^2$ так як $Q(x) = 1 * x^4 + 0 * x^3 + 1 * x^2 + 0 * x^1 + 0 * x^0$». Переходимо на крок 7.

Крок 7. Виводиться текст завдання: «На наступному кроці алгоритму побудови циклічного коду маємо виконати наступну дію:» та надається набір відповідей, представлений у таблиці 1.5.

Таблиця 1.5 – Варіанти відповідей на Кроці 7

Правильна відповідь	Відповіді
–	$Q(x)/x^r$
+	$Q(x) * x^r$
–	$Q(x) + x^r$

Якщо користувач обирає неправильну відповідь, то виводиться повідомлення «На наступному кроці алгоритму побудови циклічного коду маємо виконати наступну дію: $Q(x) * x^r$». Переходимо на крок 8.

Крок 8. Виводиться текст завдання: «Поліном $Q(x) * x^r$ має вигляд:» та надається список варіантів відповідей, представлений у таблиці 1.6.

Таблиця 1.6 – Варіанти відповідей на Кроці 8

Правильна відповідь	Відповіді
–	$Q(x) * x^4 = x^8 + x^5$
+	$Q(x) * x^4 = x^8 + x^6$
–	$Q(x) * x^4 = x^9 + x^5$
–	$Q(x) * x^4 = x^4 + x$

Якщо користувач обирає неправильну відповідь, то виводиться повідомлення: «$Q(x) * x^4 = (x^4 + x^2) * x^4 = x^8 + x^6$». Переходимо на крок 9.

Крок 9. Виводиться текст завдання: «Нехай обрано твірний поліном $P(x) = x^4 + x^3 + 1$. Поділимо $Q(x) * x^4$ на $P(x)$. Отримаємо:» та наводяться варіанти відповідей, представлені в таблиці 1.7.

Таблиця 1.7 – Варіанти відповідей на Кроці 9.

Правильна відповідь	Відповіді	
–	Частка	$x^4 + x^3 + 1$
	Остача	$x + 1$
–	Частка	$x^3 + x^2 + 1$
	Остача	x
+	Частка	$x^4 + x^3 + 1$
	Остача	1

Якщо було обрано неправильну відповідь, то виводиться повідомлення: «Рішення зображене на рисунку. Отже, частка має вигляд: $x^4 + x^3 + 1$, а остача – 1». Приклад рішення зображено на рис. 1.1.

$$
\begin{array}{l|l}
\oplus\ x^8 + x^6 & x^4 + x^3 + 1 \\
\hline
x^8 + x^7 + x^4 & x^4 + x^3 + 1 \\
\oplus\ x^7 + x^6 + x^4 & \\
\hline
x^7 + x^6 + x^3 & \\
\oplus\ x^4 + x^3 & \\
\hline
x^4 + x^3 + 1 & \\
1 &
\end{array}
$$

Рисунок 1.1 – Приклад рішення завдання для Кроку 9

Переходимо на крок 10.

Крок 10. Виводиться текст завдання: «Отримали остачу $R(x) = 1$. Записати двійковий набір, що відповідає $R(x)$:» та надається комірка для вводу відповіді. Введене значення порівнюється з правильною відповіддю: 0001. Якщо введене значення не співпадає з правильною відповіддю, то вивести повідомлення: «Кількість розрядів, у комбінації, що відповідає остачі $R(x)$ дорівнює: $r = 4,\ R(x) = 0 * x^3 + 0 * x^2 + 0 * x^1 + 1 * x^0 = 1$. Таким чином вірна комбінація: 0001». Переходимо на крок 11.

Крок 11. Виводиться текст завдання: «Введіть кодову комбінацію двійкового циклічного коду:» та надається комірка для вводу відповіді. Введене значення порівнюється з правильною відповіддю 101000001. Якщо введене значення не співпадає з правильною відповіддю, то вивести повідомлення: «Для утворення комбінації двійкового циклічного коду треба задану комбінацію 10100 доповнити комбінацією для $R(x)$ 0001, тобто правильна відповідь: 101000001».

Алгоритм завершено.

1.2 Вимоги до тренажера

Спираючись на наведений приклад у розділі 1.1 можна сформувати наступні функціональні вимоги, які повинні бути реалізовані в тренажері:

– надання теоретичного матеріалу з теми «Коди із виправленням помилок»;

– надання можливості виконання різноманітних завдань та тестування, зокрема: повторення теоретичного матеріалу, виконання практичних завдань та тестування студентів;

– надання можливості виконання наступних типів завдань: завдання з однією правильною відповіддю, завдання з декількома правильними відповідями, завдання по співставленню відповідей та завдання з полем для введення відповіді;

– надання можливості створювати практичні задачі та тести за допомогою зручного інтерфейсу та зберігати їх в переносимому форматі, також має бути можливість редагувати вже існуючі завдання та завантажувати їх з файлів;

– при вирішенні завдань тренажер повинен надавати «другий шанс» під час виконання завдань, а також надавати підказки і пояснення при неправильних відповідях;

– в кінці виконання усіх завдань тесту або задачі тренажер повинен надавати стислу статистику результатів виконання.

До нефункціональних вимог можна віднести наступні:

– зручність та зрозумілість інтерфейсу користувача;

– наявність інструкції користувача;

– швидкість та стабільність роботи тренажеру;

– відкритість програмного коду, що дозволить підтримувати тренажер у майбутньому.

2 ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Огляд робіт, де розглянуте аналогічне до теми роботи завдання

Незважаючи на стрімкий розвиток інформаційних технологій та мережі Інтернет, розвиток у сфері навчання є дуже повільним та майже не привносить нові методи та можливості. Навіть події 2020 року та використання дистанційної форми навчання майже не призвело до створення нових методів та вдосконалення старих. Так, наприклад, деякі зі створених тренажерів були створені ще на початку розвитку Інтернету та не вдосконалювались з того часу. Все це, а також специфічність теми роботи призвела до відсутності будь-яких схожих програмних засобів за обраною темою. Тому буде проведено огляд схожих робіт з інших галузей.

Світова тенденція на використання дистанційної форми навчання з'явилась тільки декілька років назад, хоча сама ідея – не нова. Одними з найпопулярніших платформ на сьогодні є: Coursera, edX, Udemy, Khan Academy, Google Classroom, Cisco Networking Academy та багато інших, зображені на рис. 2.1 – 2.3 [4].

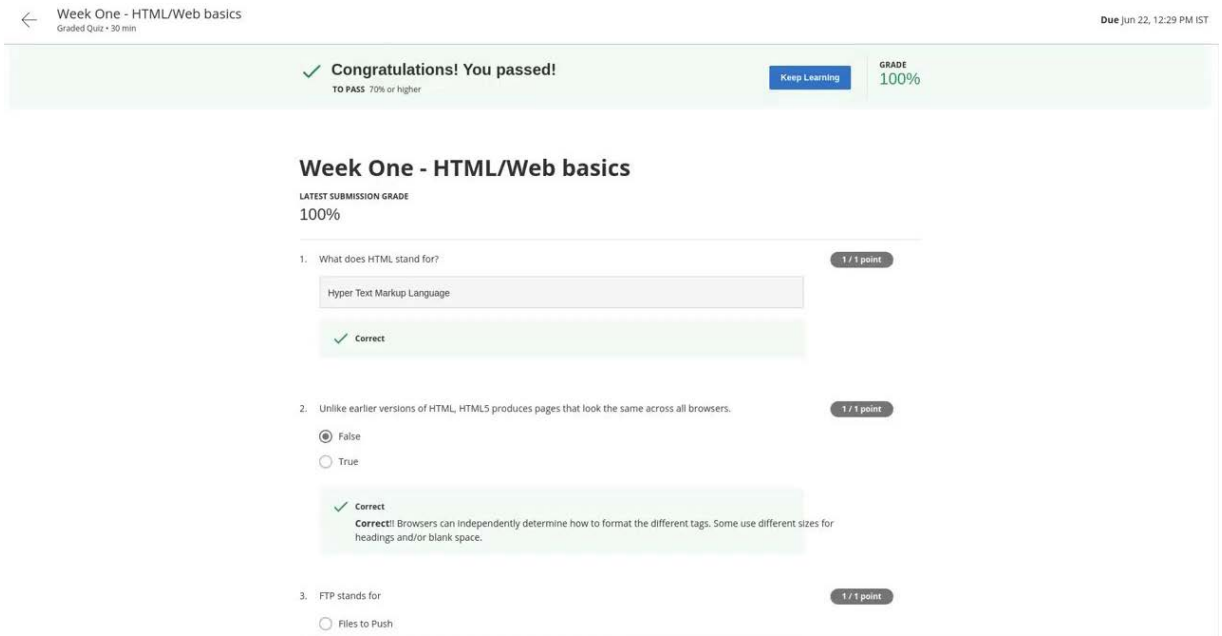

Рисунок 2.1 – Приклад роботи платформи Coursera

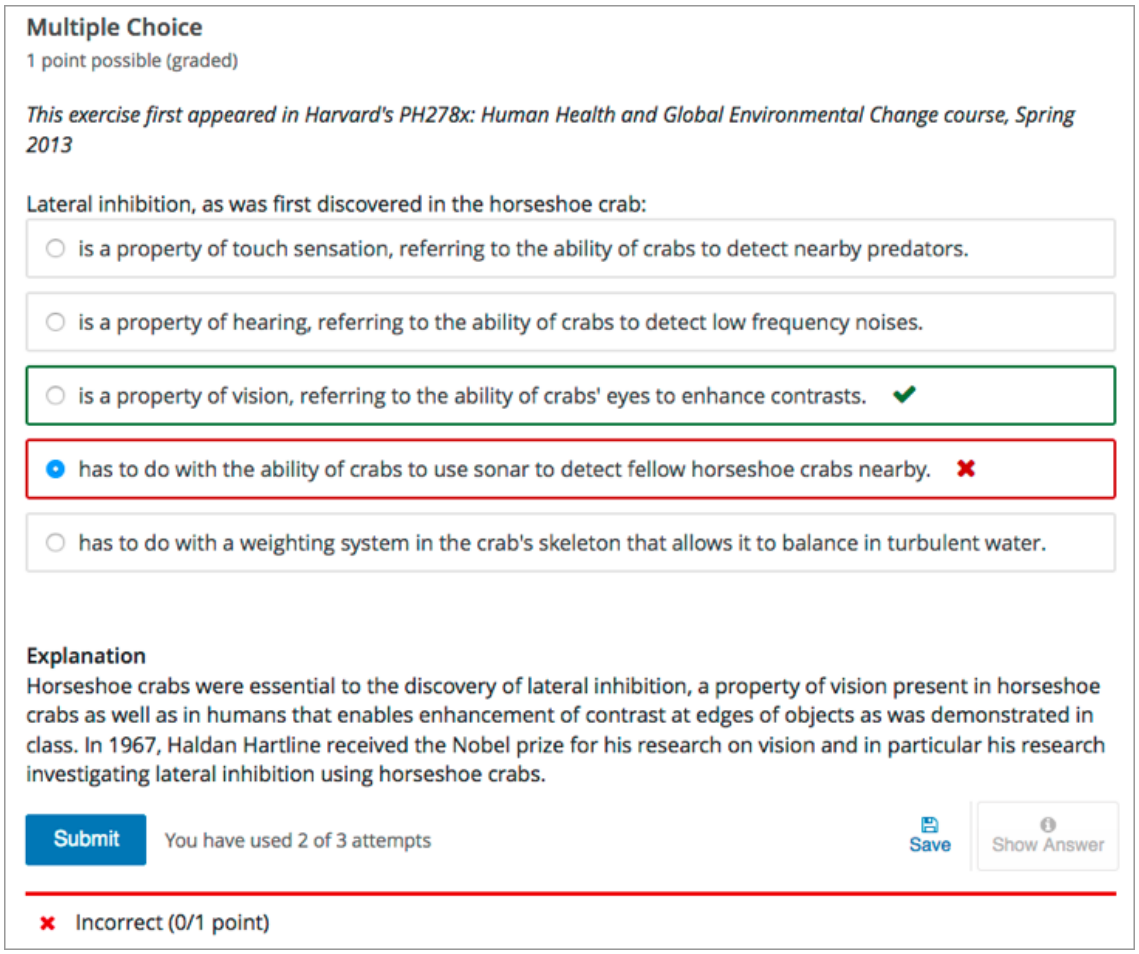

Рисунок 2.2 – Приклад роботи платформи edX

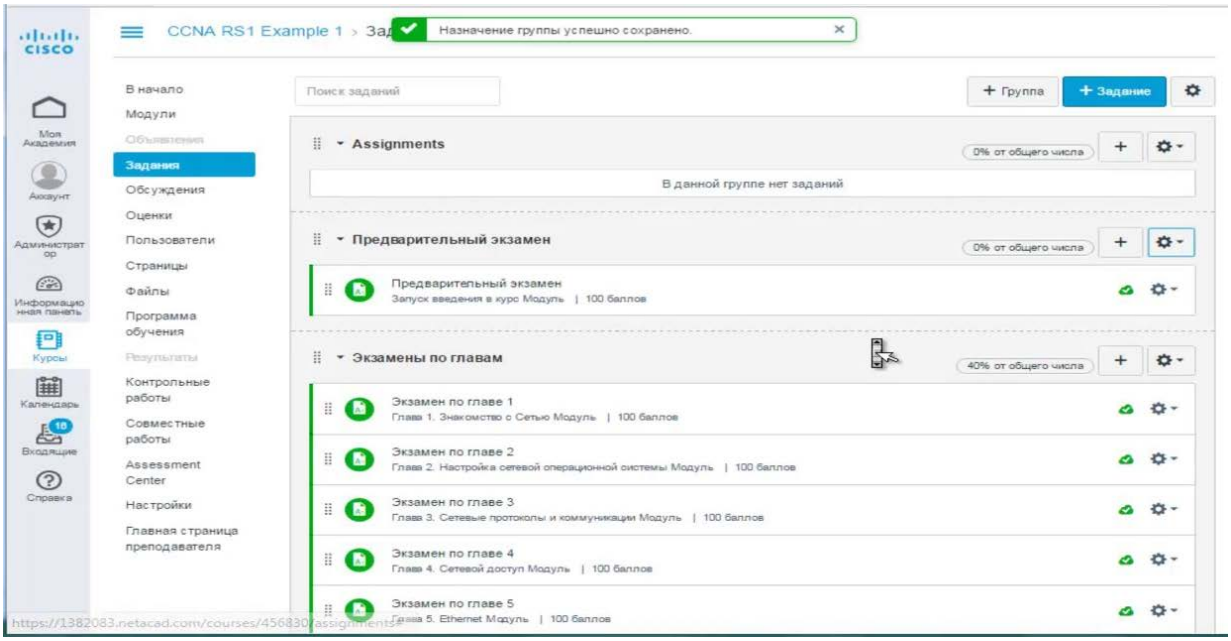

Рисунок 2.3 – Приклад роботи платформи Cisco Networking Academy

В Україні популярними платформами є Moodle, зображена на рис 2.4, та Prometheus [5]. Усі ці платформи є веб-застосунками або веб-сервісами, які надають велику кількість різноманітних навчальних курсів здебільшого на безкоштовній основі, проте для отримання персонального сертифікату потрібно оплатити курс.

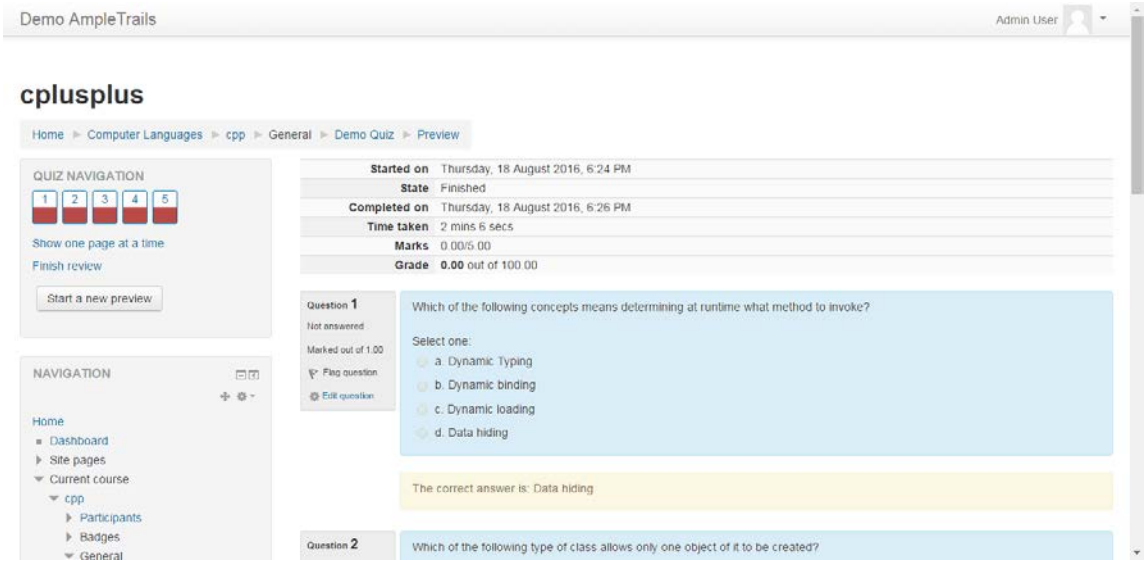

Рисунок 2.4 – Приклад роботи платформи Moodle

Протягом навчання студент має переглядати відеолекції, які надсилаються йому щотижня (відео найчастіше діляться на частини тривалістю 15-20 хвилин), читати рекомендовані статті, виконувати домашні завдання. Домашні завдання бувають у вигляді тестів, написання есе, творчих завдань чи проектів. В той час, як в тестах потрібно надати усі відповіді, після чого тест буде оцінено автоматично, для оцінки виконання інших типів завдань, наприклад в Coursera, розроблено технологію, за якою студенти оцінюють та коментують роботи один одного [6].

На відміну від інших сервісів Moodle та Google Classroom надають можливості організації власних курсів. Так наприклад, типова функціональність Moodle включає в себе [7]:

– здача завдань та оцінювання та онлайн тестування;
– дискусійні форуми та обмін повідомленнями;

– завантаження файлів;

– календар подій, новини та анонси подій;

– вікі.

З іншої сторони існують програми та мобільні додатки, які на відміну від веб-платформ, орієнтовані більше на вирішення конкретних тестів та завдань, а ніж на організацію навчання в цілому. Проте, хоча таких додатків багато, тих додатків як б саме навчали дуже мало та майже всі вони навчають за більш загальними темами такими, як математика, іноземні мови та багато іншого. Серед них можна виділити такі програми, як SoloLearn – орієнтовану на вивчення різних мов програмування, зображену на рис. 2.5 [8], та Duolingo – орієнтовану на вивчення іноземних мов, зображену на рис. 2.6 [9]. Обидві програми є мобільними додатками та є тренажерами в класичному розумінні цього слова.

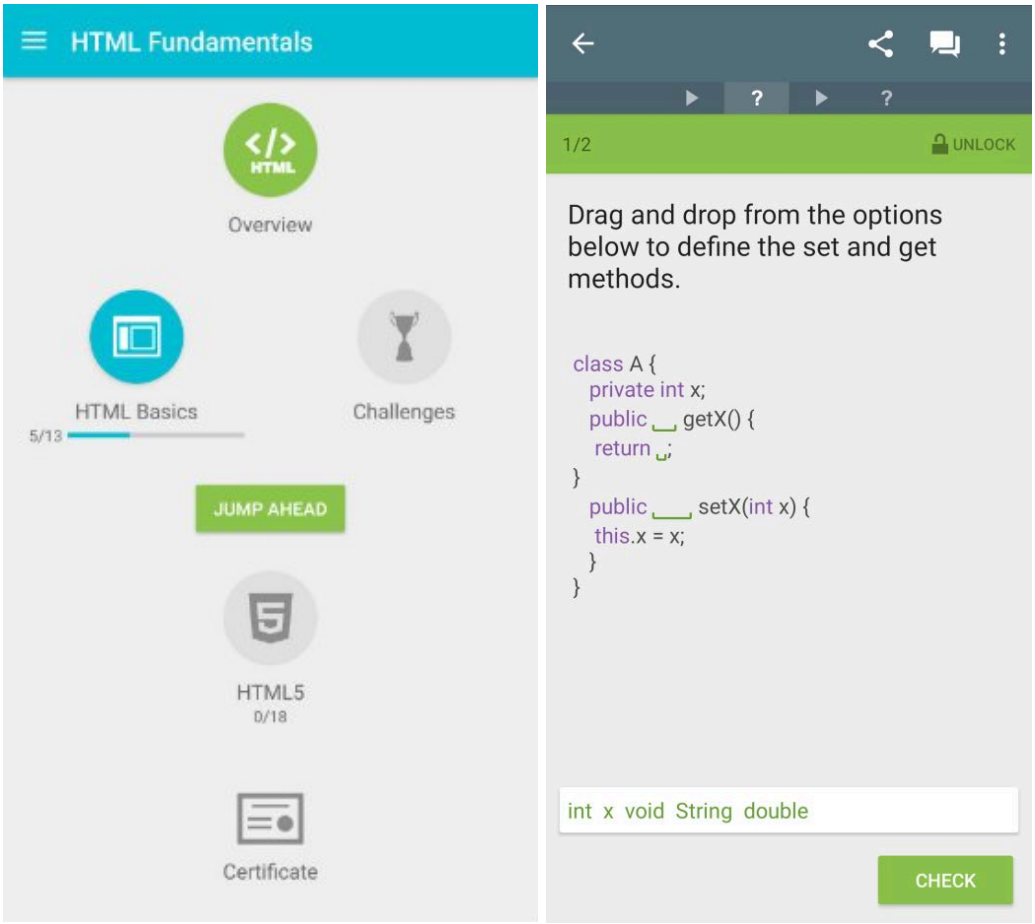

Рисунок 2.5 – Приклад роботи застосунку SoloLearn

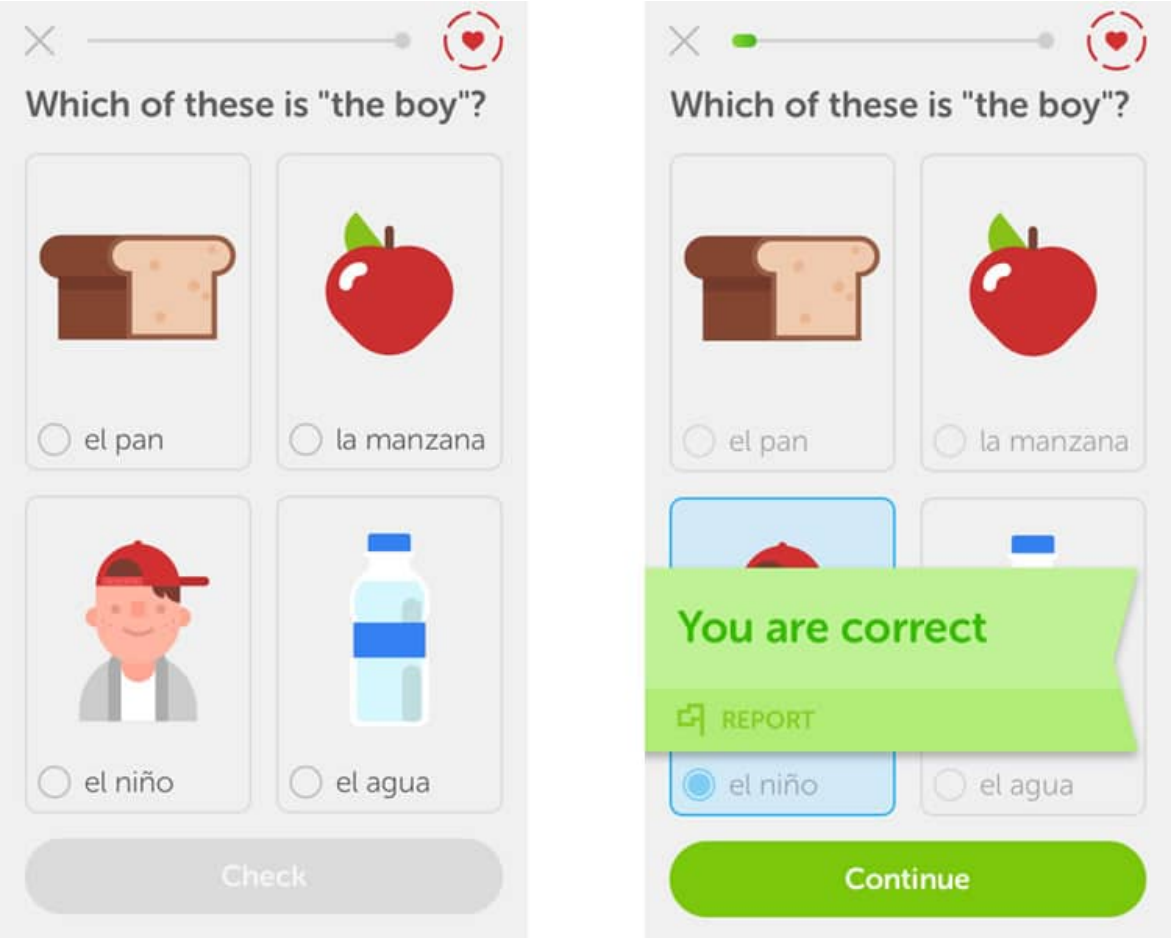

Рисунок 2.6 – Приклад роботи застосунку Duolingo

Так наприклад, SoloLearn, перед тестуванням, надає короткі відомості за темою, що вивчається, а потім дає невеликий тест з різноманітними завданнями, при цьому якщо завдання виконано неправильно, то про це одразу буде відомо. В Duolingo все майже так само, як і в Sololearn, проте відомостей за темою надається ще менше або ніяких в загалі.

Також варто зазначити про існування великої кількості класичних навчальних тренажерів з різних тем, які було створено 10-20 років тому назад. Хоча деякі з них існують навіть сьогодні, їх дуже складно знайти, бо здебільшого вони розроблялись для використання в межах навчального закладу, і навіть запустити. З боку функціональності ці тренажери були примітивними та орієнтовані тільки на перевірку знань студентів.

2.2 Позитивні аспекти оглянутих робіт

З позитивних аспектів оглянутих сервісів можна відзначити наступні:

– здебільшого безкоштовність більшої частини функціоналу;

– наявність зручного інтерфейсу користувача;

– кросплатформеність та наявність веб-версії в багатьох з сервісів, що дає можливість використовувати сервіси будь-де, де є інтернет;

– можливість збереження результатів та повторного проходження;

– можливість організовувати навчальний процес в деяких з сервісів;

– популярність більшості, що дозволяє стверджувати, що їх навряд покинуть протягом наступних років.

2.3 Вади розробок з оглянутих робіт

Не зважаючи на популярність даних веб сервісів та додатків та їх переваги вони мають велику кількість недоліків:

– всі сервіси, окрім Moodle, є ПЗ з закритим початком кодом, що не дозволяє модифікувати ці сервіси під свої потреби;

– всі сервіси та додатки є не повністю безкоштовними, що не дає можливості отримати весь потрібний функціонал;

– більшість з оглянутих аналогів спеціалізується на якійсь одній темі навчання та не має потрібного функціоналу, який би надав можливість створити завдання з іншої теми;

– майже всі сервіси орієнтовані на перевірку знань користувачів за темою, а не навчання за нею, особливо це стосується онлайн платформ, де проходить перевірка теоретичного матеріалу за темою та майже відсутні практичні завдання. В той самий час програми-тренажери, такі як SoloLearn, майже не надають повних теоретичних відомостей з теми та перевіряють знання не на реальних задачах, з якими може зіткнутися майбутній спеціаліст;

– велика кількість програм-тренажерів, які створюються в вигляді мобільних застосунків, розробляються не викладачами та здебільшого позиціонують себе як розважальні програми, а ніж як навчальні програми.

2.4 Необхідність та актуальність теми роботи

Підсумовуючи все вище згадане, можна зробити наступні висновки про ситуацію, що склалась в галузі навчання:

– великі онлайн-платформи для навчання орієнтовані: або на організацію процесу навчання в цілому або тільки на подачу теоретичного матеріалу та проведення тестування з нього, при цьому навчання з точки зору практичної частини залишається все так само в «ручному режимі», що при збільшенні кількості студентів потребує більшої кількості викладачів для підтримання однакової якості освіти у всіх студентів;

– невеликі тренажери, які існують в якості мобільних додатків, показують іншу тенденцію, а саме орієнтованість на невеликі практичні тести, а не на реальні задачі, та майже цілковиту відсутність теоретичного матеріалу, це все призводить до ще більшого погіршення знань в тих, хто навчається;

– загальною проблемою як тренажерів, так і онлайн-платформ є відсутність багатьох тем для навчання, зокрема з теми «Коди із виправленням помилок», або інструментів для створення матеріалів за цими темами, що негативно впливає на якість підготовки фахівців.

В той же самий час із збільшенням кількості інформації та запровадженням нових технологій росте потреба в висококваліфікованих фахівцях із знаннями «Теорії інформації та кодування».

Саме тому виникає потреба в створені програми-тренажеру з теми «Коди із виправленням помилок», яка дозволить спростити та пришвидшити навчання не тільки з теоретичних питань, а також допоможе в автоматизації навчання при вирішенні практичних задач з теми.

3 ТЕОРЕТИЧНА ЧАСТИНА

3.1 Алгоритмізація задачі за темою роботи

Спираючись на приклад та вимоги до тренажеру, описані у розділі 1, можна сформувати наступний алгоритм, який підходить для рішення більшості задач з теми «Коди із виправленням помилок» дистанційного навчального курсу «Теорія інформації і кодування».

Початок алгоритму.

Крок 1. Завантаження, з файлу, даних про задачу, а саме: її назва, опис, складність, кількість завдань та дані про кожне завдання, де кожне завдання складається з: запитання, відповідей до нього з позначенням правильних відповідей, підказки, пояснення, а також додаткові матеріали до запитання та пояснення у вигляді графічних матеріалів.

Крок 2. Якщо номер завдання не перевищує кількості завдань у задачі, то переходимо до кроку 3, інакше переходимо до кроку 10.

Крок 3. Визначаємо тип завдання, якщо тип завдання це:

– завдання по співставленню відповідей, то переходимо на крок 4;

– завдання з полем для введення відповіді, то переходимо на крок 5;

– завдання з однією правильною відповіддю, то переходимо на крок 6;

– завдання з декількома правильними відповідями, то переходимо на крок 7.

Крок 4. На екран виводиться набор запитань та на набор відповідей на них. Кожне запитання потрібно зіставити з однією з відповідей, при цьому кожне запитання має тільки одну правильну відповідь, а кожна відповідь не обов'язково має запитання. Переходимо на крок 8.

Крок 5. На екран виводиться запитання та поле для вводу відповіді, в яку потрібно вписати правильну відповідь. Переходимо на Крок 8.

Крок 6. На екран виводиться запитання та варіанти відповідей до нього, серед яких потрібно обрати тільки одну правильну відповідь. Переходимо на крок 8.

Крок 7. На екран виводиться запитання та варіанти відповідей до нього, серед яких потрібно обрати декілька правильних відповідей. Переходимо на крок 8.

Крок 8. Якщо було обрано неправильну відповідь, то на екран виводиться повідомлення про помилку. Якщо в завданні є підказка, то надається можливість повторного вибору. Якщо було обрано неправильну відповідь вдруге, або в завданні немає підказки, то виводиться повідомлення про помилку та якщо у завданні є роз'яснення, виводиться повідомлення з роз'ясненням, а також відзначається яка з відповідей є правильною. Переходимо на крок 9.

Крок 9. Зберігається інформація про зроблену відповідь, з екрану прибираються наявні дані про поточне завдання, номер завдання збільшується на одиницю. Переходимо на крок 2.

Крок 10. На екран виводяться результати виконання задачі, а саме інформація про кількість правильних та неправильних відповідей, а також кількість завдань виконаних з другої спроби.

Кінець алгоритму.

3.2 Розробка блок-схеми, яка підлягає програмуванню

Алгоритм рішення задач за темою «Коди із виправленням помилок» дистанційного навчального курсу «Теорія інформації і кодування», розроблений у розділі 3.1, представлений у вигляді блок-схеми, яка підлягає програмуванню, на рис 3.1.

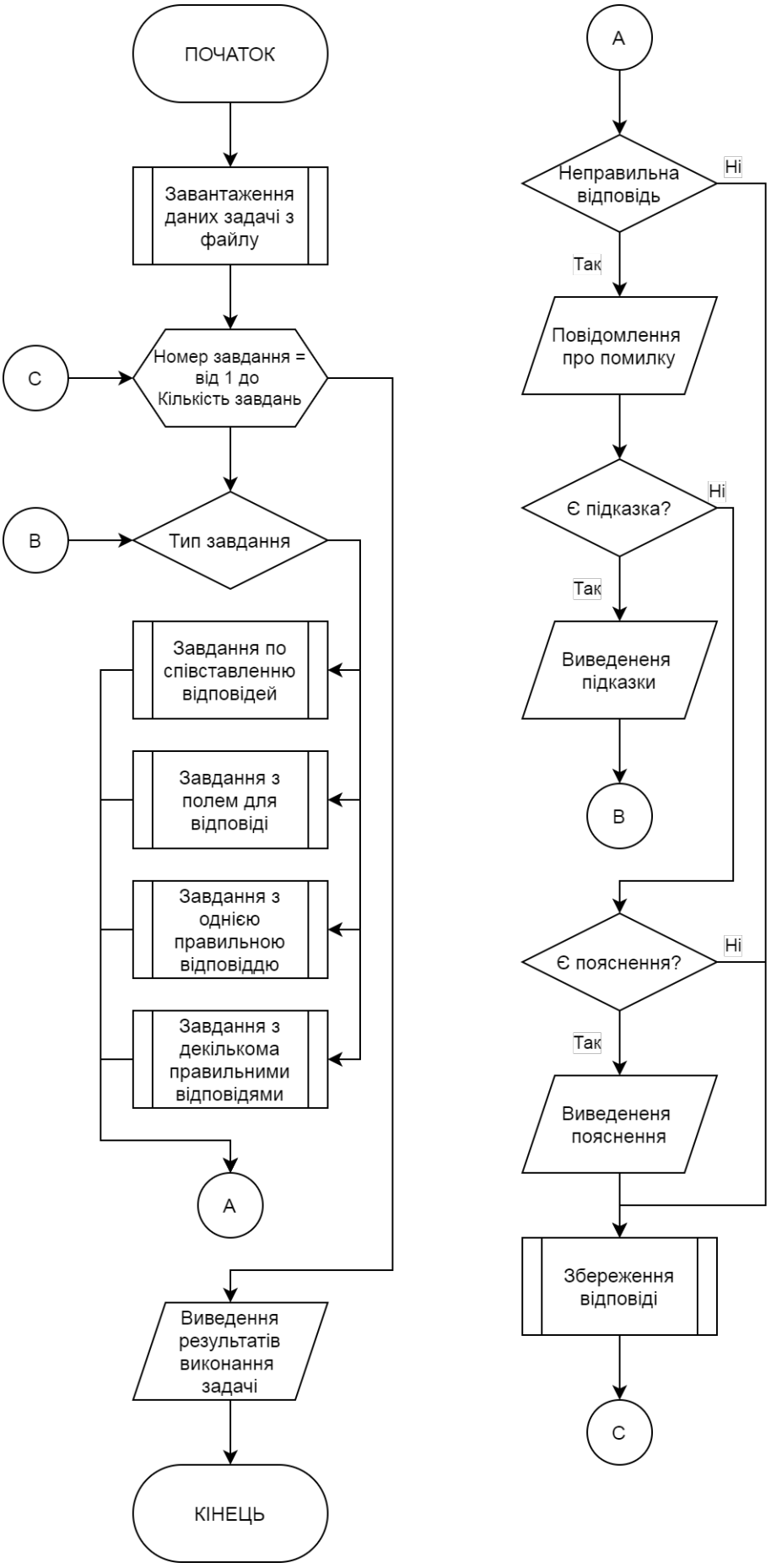

Рисунок 3.1 – Блок-схема загального алгоритму

3.3 Обґрунтування вибору програмних засобів

Спираючись на вимоги до ІС та алгоритм дій при виконанні завдання, було вирішено обрати мову програмування Object Pascal. Її багаті можливості дозволяють створювати швидке та невибагливе до ресурсів комп'ютера програмне забезпечення, а простота та зрозумілість дозволяють швидко орієнтуватись в програмному коді навіть програмістам-початківцям, що сприяє довгій підтримці ПЗ.

ОР містить усі сучасні мовні засоби, які можуть бути потрібними під час розробки ПЗ [10]:

- класи та інтерфейси;
- перевантаження операцій;
- обробка винятків;
- узагальнені класи та підпрограми;
- «збирання мусору»;
- лямбда-вирази;
- посилання і оператори управління вільно розподіленою пам'яттю та багато іншого.

Також особливостями мови Object Pascal можна назвати «сильну» типізацію та мінімальну кількість синтаксичних неоднозначностей, при чому сам синтаксис є інтуїтивно зрозумілим навіть при першому знайомстві з мовою. Наявність вільних реалізацій забезпечила широку переносимість програм на різні платформи. Усі ці особливості дають можливість конкурувати з такими мовами, як C++, Java та C#, де ці особливості реалізовані частково або невдало.

В той самий час в ОР, як і в багатьох інших мовах програмування, відсутня стандартна бібліотека для реалізації ПЗ з графічним інтерфейсом користувача. Проте, завдячуючи багатій історії Object Pascal та великій

спільноті відкритого програмного забезпечення, існує багато відкритих реалізацій для створення застосунків з графічним інтерфейсом користувача.

Для реалізація обраної ІС було обрано IDE Lazarus – вільне середовище розробки ПЗ для компілятора Free Pascal Compiler. Інтегроване середовище розробки надає можливість багатоплатформової розробки застосунків, в тому числі з графічним інтерфейсом. Головними перевагами Lazarus є [11]:

– велика кількість графічних компонентів, які просто та зручно налаштовувати;

– вбудований налагоджувач;

– повністю юнікодний (UTF-8) інтерфейс і редактор, тому відсутні проблеми з портуванням коду, що містить національні символи;

– потужний редактор коду, що включає систему підказок, гіпертекстову навігацію по вихідних текстах, автозавершення коду і рефакторинг;

– форматування коду «з коробки», використовуючи механізми Jedi Code Format;

– має власний формат керування пакунками;

– автозбирання самого себе (під нову бібліотеку віджетів) натисненням однієї кнопки;

– Підтримувані для компіляції ОС: Linux, Microsoft Windows (Win32, Win64), Mac OS X, FreeBSD, WinCE, OS/2.

4 ПРАКТИЧНА ЧАСТИНА

4.1 Опис процесу програмної реалізації

Процес програмної реалізації – це сукупність ряду послідовних дій, спрямованих на розробку програмного забезпечення [12]. Процес розробки складається з безлічі підпроцесів, в деяких моделях, наприклад водоспаду, вони йдуть один за одним, в інших моделях їх порядок або склад може змінюватись. Головними кроками в будь-якій моделі є [13]:

- аналіз вимог;
- проектування програмного забезпечення;
- програмування;
- тестування програмного забезпечення;
- системна інтеграція;
- впровадження програмного забезпечення;
- супровід програмного забезпечення.

Існує декілька моделей процесу розробки ПЗ: водоспадна (каскадна), ітераційна, спіральна, гнучка, та багато інших. Кожна з них має свої переваги та недоліки. Тому зазвичай модель процесу розробки програмного забезпечення обирають в залежності від поставлених до проекту вимог, щоб максимально використати переваги того чи іншого підходу.

Виходячи з вимог до розроблюваного ПЗ було вирішено обрати ітераційну модель розробки, яку зображено на рис. 4.1. Ітераційна (ітеративна та інкрементна розробка) – це серцевина циклічного процесу розробки ПЗ, який був розроблений у відповідь на слабкі сторони водоспадної моделі [14]. Процес починається з початкового планування і закінчується впровадженням (готового ПЗ) з циклічними взаємодіями між цими етапами.

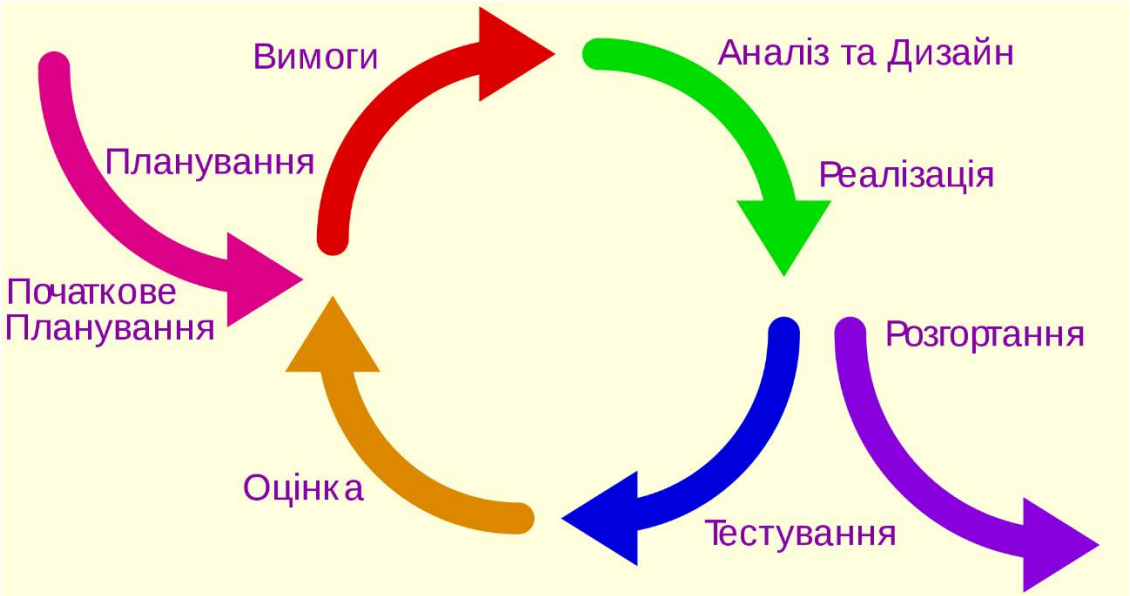

Рисунок 4.1 – Ітераційна модель розробки

Інкрементна розробка поділяє систему на інкременти (порції). У кожній порції доставляється певна частина функціональності. Уніфікований процес поділяє порції/ітерації на такі фази: початок, створення плану, побудова та перехід [15]:

– початок визначає масштаб проєкту, ризики і вимоги (функціональні та нефункціональні) на високому рівні, але достатньому для того, щоб оцінити складність;

– результатом створення плану стає виробнича архітектура, яка пом'якшує найзначніші ризики та заповнює нефункціональні вимоги;

– побудова поступово (інкрементно) заповнює архітектуру готовим кодом, який створено завдяки аналізу, плануванню, реалізації та тестуванню функціональних вимог;

– перехід переводить систему у виробниче середовище.

Кожна з фаз може бути розділена на 1 або більше ітерацій, які, зазвичай, обмежені у часі, а не у функціональності. Архітектори та аналітики працюють на одну ітерацію попереду програмістів та тестувальників, для того, щоб тримати їхній перелік завдань повним.

Перевагами ітеративного підходу є [16]:

– зниження впливу серйозних ризиків на ранніх стадіях проекту, що веде до мінімізації витрат на їх усунення;

– організація ефективного зворотного зв'язку проектної команди зі споживачем (а також замовниками,) і створення продукту, який реально відповідає його потребам;

– акцент зусиль на найбільш важливі і критичні напрямки проекту;

– безперервне ітеративне тестування, що дозволяє оцінити успішність всього проекту в цілому;

– раннє виявлення конфліктів між вимогами, моделями і реалізацією проекту;

– більш рівномірне завантаження учасників проекту;

– ефективне використання накопиченого досвіду;

– реальна оцінка поточного стану проекту і, як наслідок, велика впевненість замовників і безпосередніх учасників в його успішному завершенні;

– витрати розподіляються по всьому проекту, а не групуються в його кінці.

4.2 Опис програми

Завдяки використанню мови ОР структура програми є логічною завдяки тому, що логічно пов'язані між собою елементи розташовані в різних програмних модулях, де кожному програмному модулю відповідає інформація про відповідну графічну форму та розташованих на ній елементах. Інформація, яка повторюється в багатьох модулях та яку не можливо пов'язати з конкретною графічною формою знаходиться в окремому програмному модулі, а саме інформація про задачу, її збереження у файл та завантаження з нього. Це дозволяє швидко реагувати на можливі помилки в програмі.

Кожне вікно програми складається з форми вікна та елементів розташованих на ній, при цьому взаємодія між користувачем та інтерфейсом відбувається за допомогою системи подій, коли кожен елемент інтерфейсу викликає функцію-обробник для тієї чи іншої події, наприклад для обробки натискання клавіши. Форми вікон програми з розташованими на них елементами зображені на рис. 4.2 - 4.11.

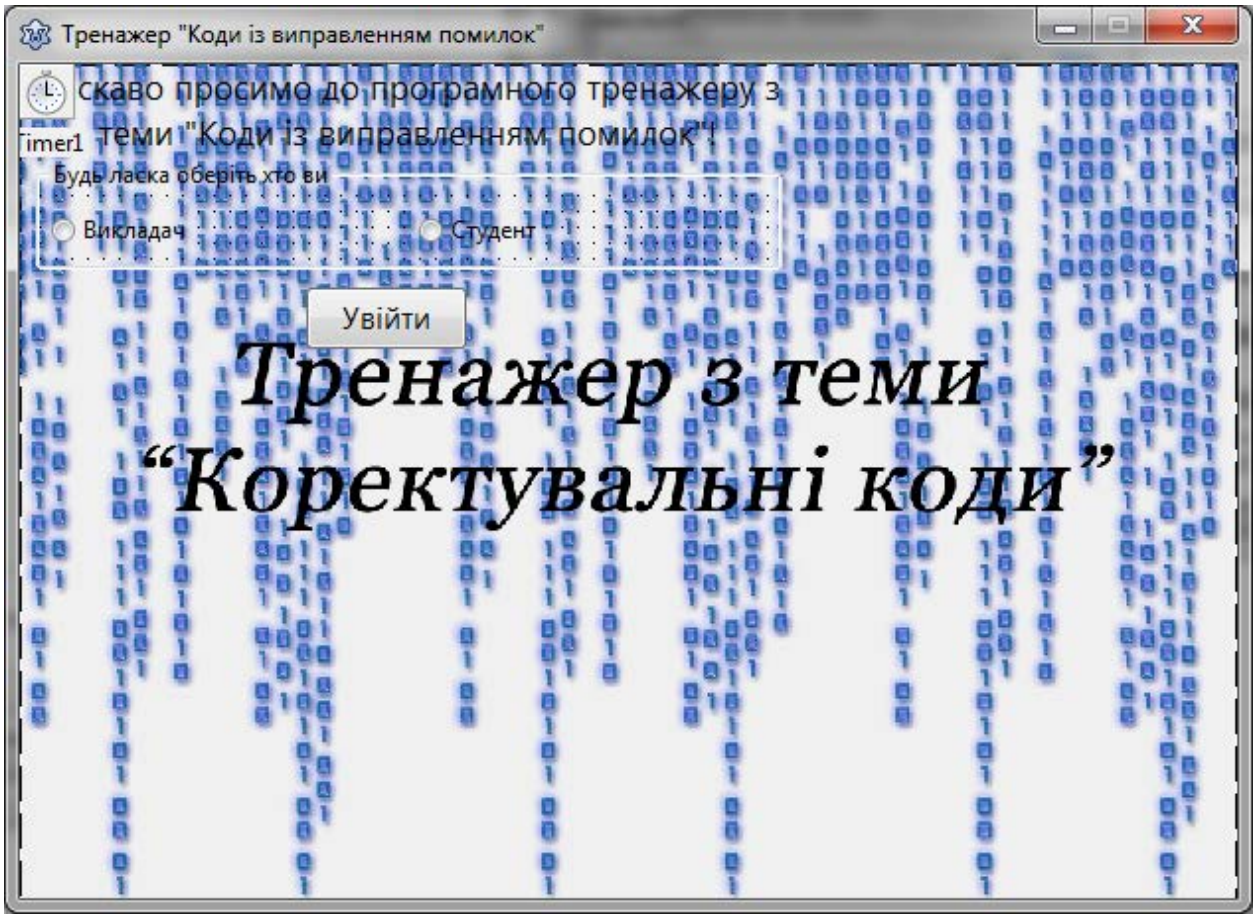

Рисунок 4.2 – Форма головного вікна програми

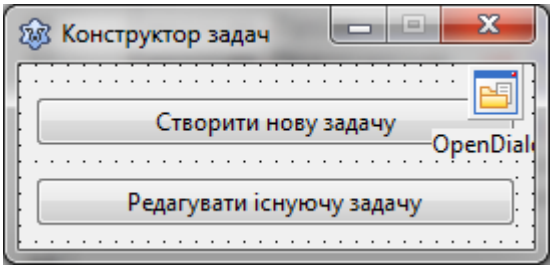

Рисунок 4.3 – Форма вікна дій користувача з обліковим записом «Викладач»

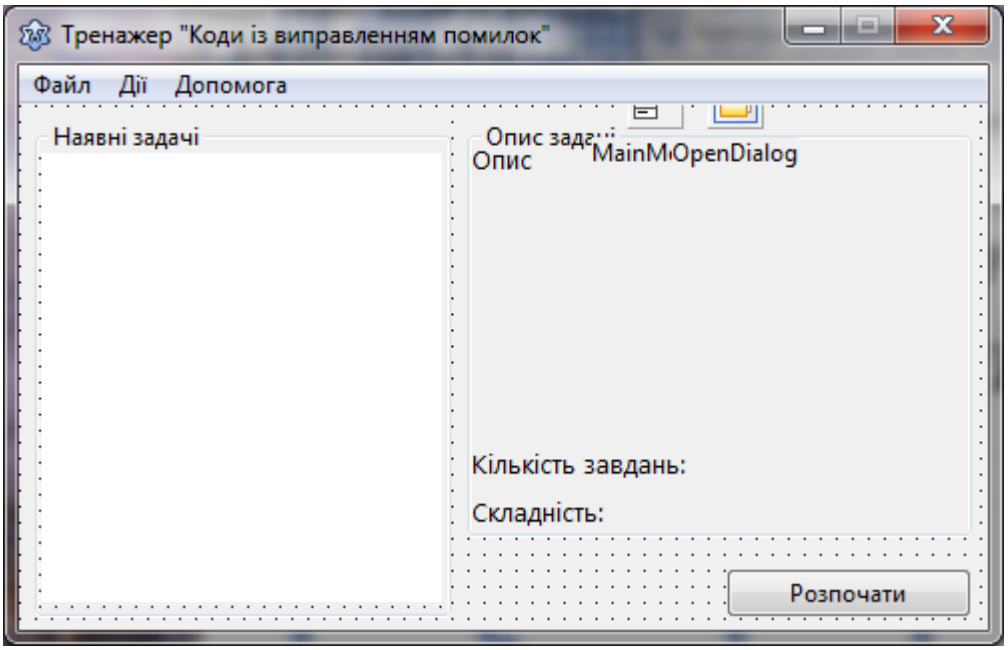

Рисунок 4.4 – Форма вікна дій користувача з обліковим записом «Студент»

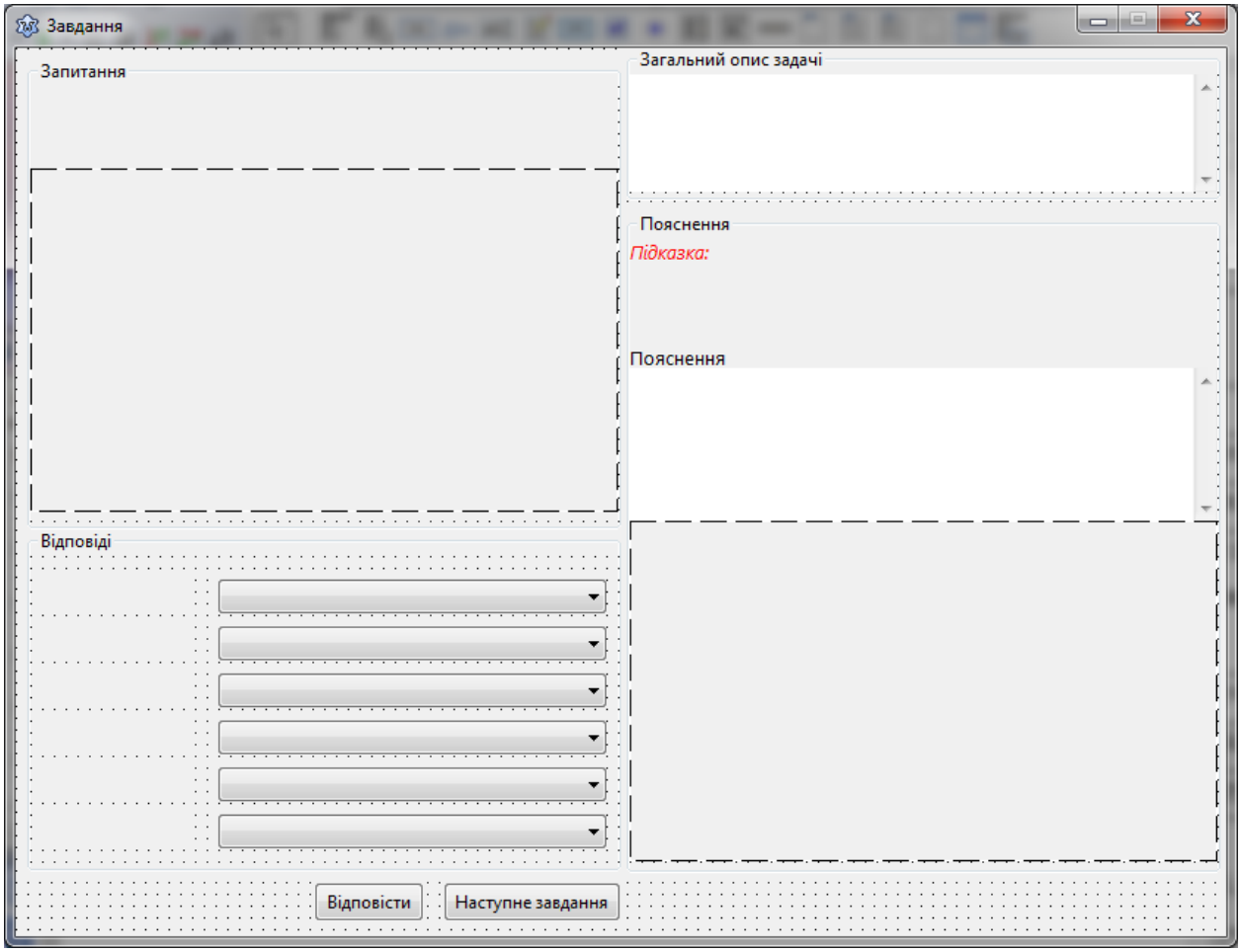

Рисунок 4.5 – Форма вікна вирішення задачі

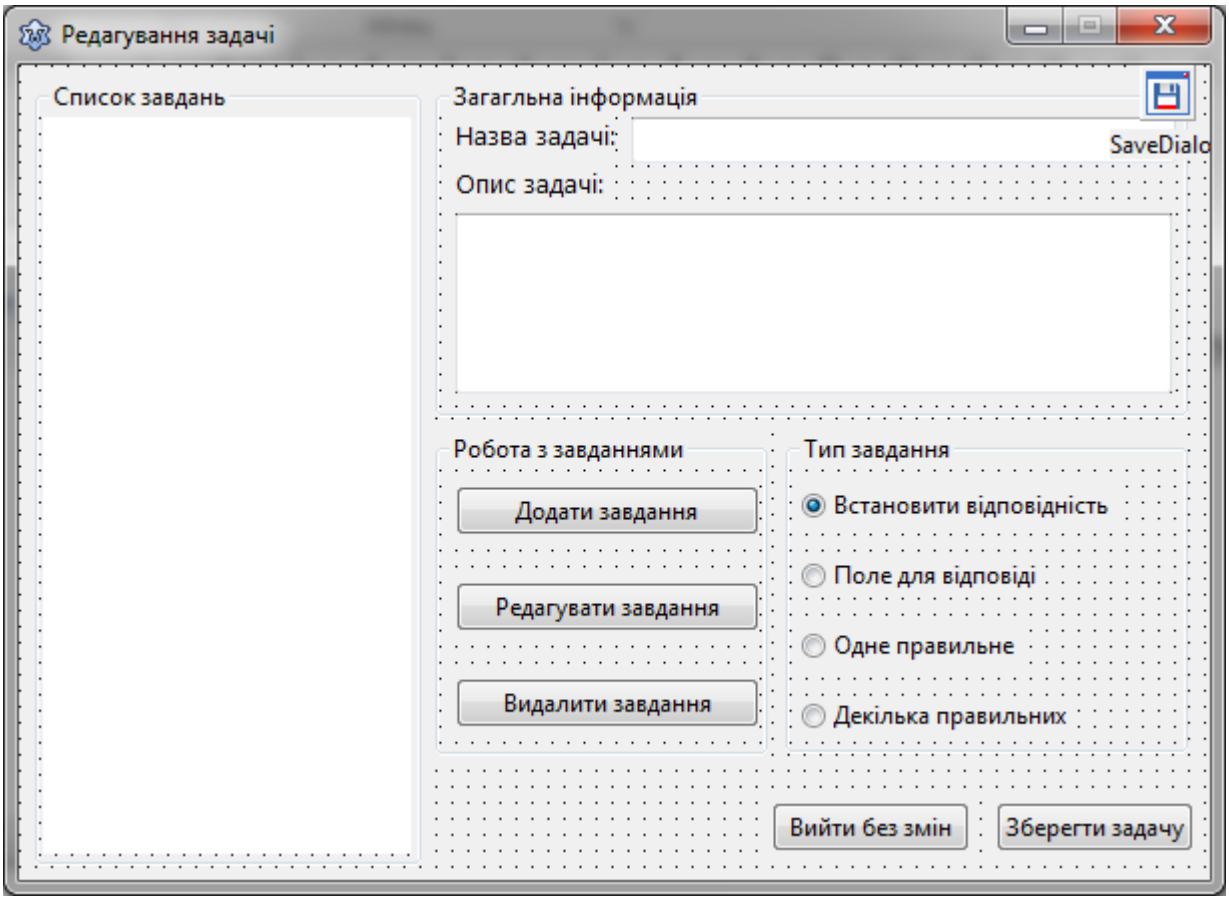

Рисунок 4.6 – Форма вікна створення/редагування задачі

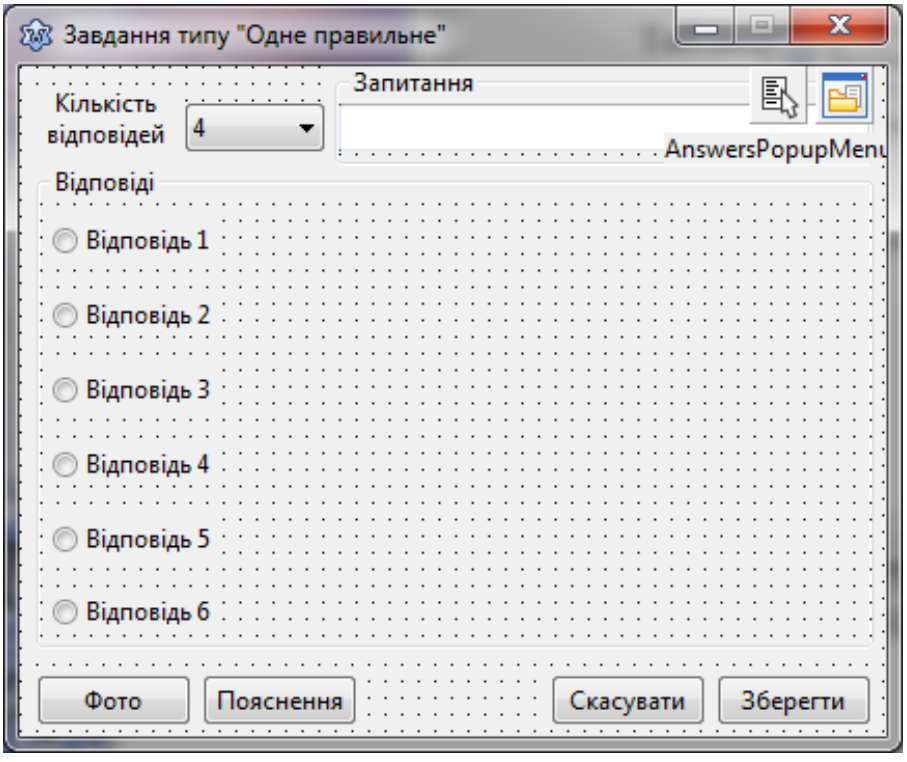

Рисунок 4.7 – Форма вікна створення/редагування завдання з однією відповіддю

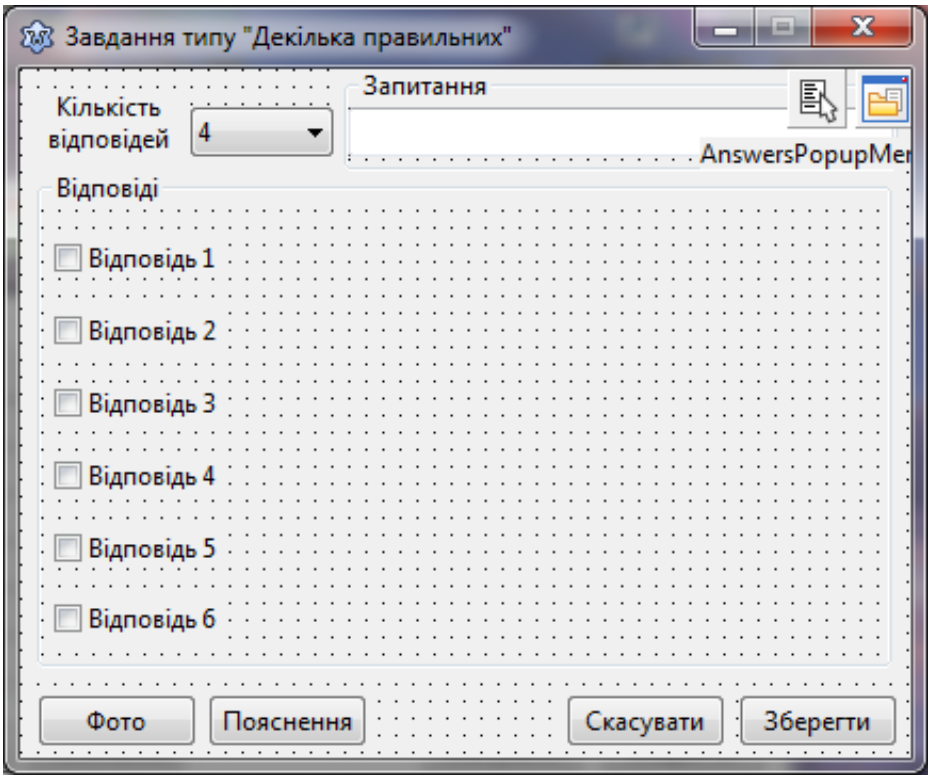

Рисунок 4.8 – Форма вікна створення/редагування завдання з декількома відповідями

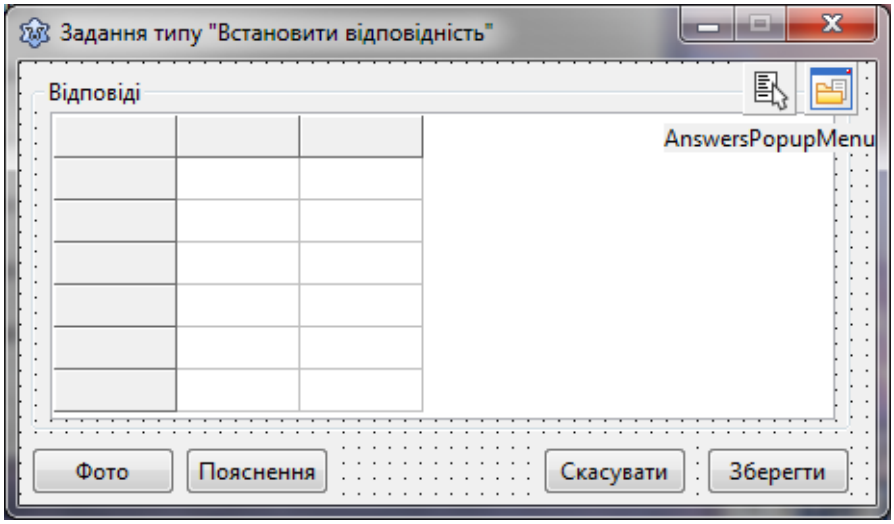

Рисунок 4.9 – Форма вікна створення/редагування завдання зі співставлення відповідей

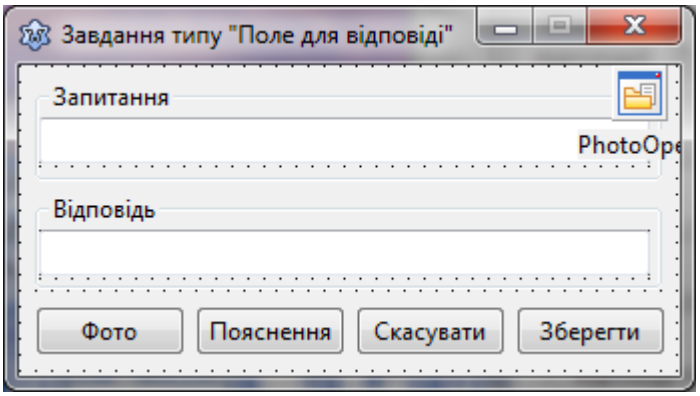

Рисунок 4.10 – Форма вікна створення/редагування завдання з полем для відповіді

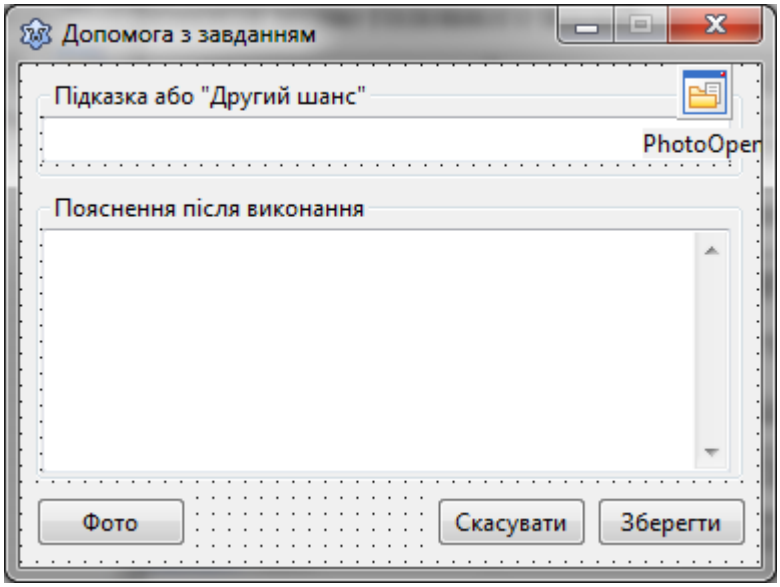

Рисунок 4.11 – Форма вікна створення/редагування підказки та пояснення до завдання

4.3 Тестування програми

Для перевірки правильності роботи створеного тренажеру, було проведено його тестування. В якості задачі, для тестування можливостей тренажеру, було обрано приклад задачі, розглянутої у розділі 1.1.

Після запуску задачі на виконання, з'явиться вікно з першим завданням, яке є завданням на співставлення відповідей, якщо одна з відповідей буде не

привильною, то повинно з'явитись пояснення до завдання та повинні бути виведені правильні відповіді. Результати виконання першого завдання, зображені на рис. 4.12.

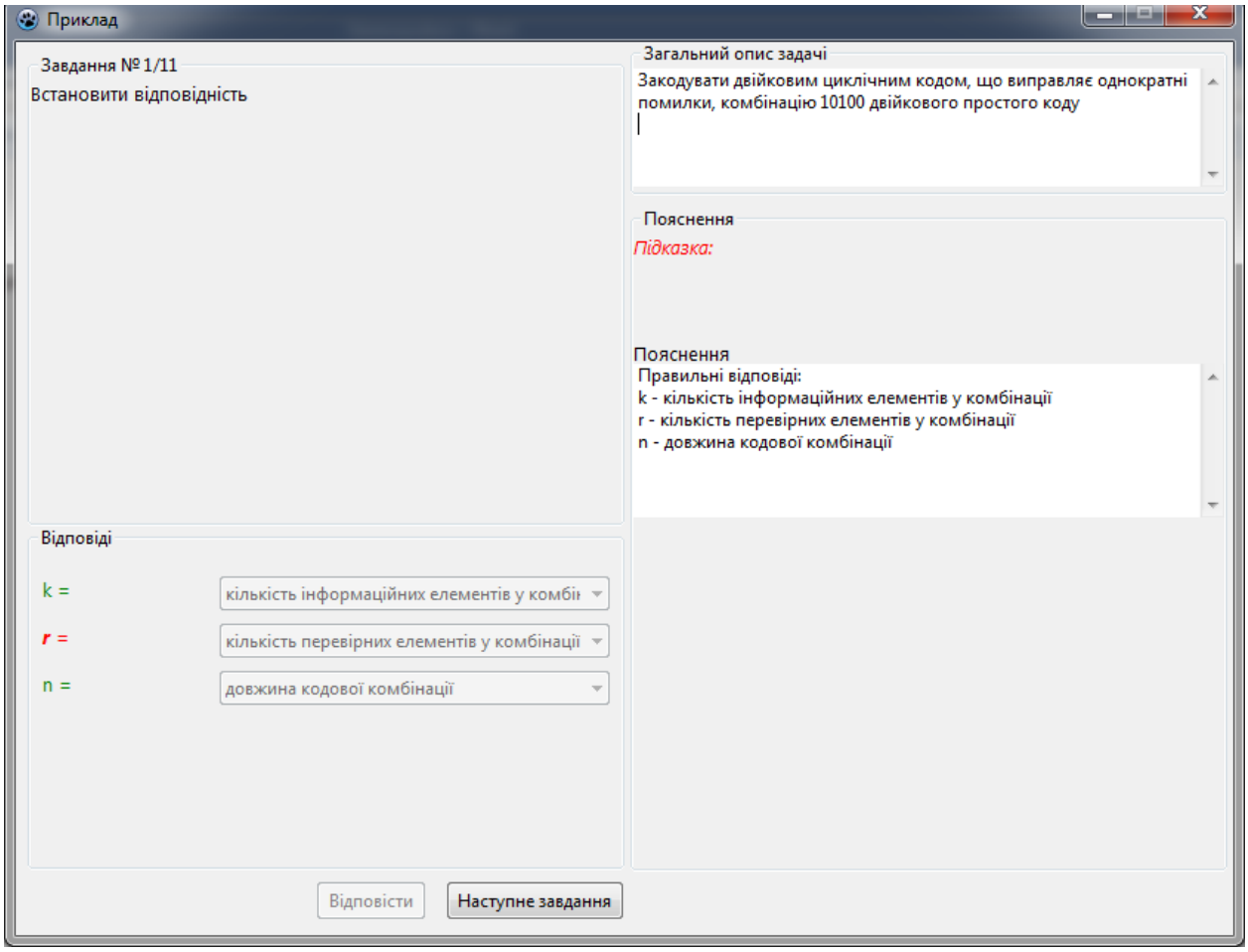

Рисунок 4.12 – Результати виконання першого завдання

Друге завдання є завданням з однією правильною відповідю. Згідно прикладу, якщо буде зроблено неправильну відповідь, то має з'явитись пояснення, та буде обрано правильну відповідь. Якщо буде зроблено правильну відповідь, то буде виведно відповідне повідомлення, інакше буде виведено повідомлення про помилку та пояснення до виконання завдання. Результати виконання другого завдання, зображені на рис. 4.13.

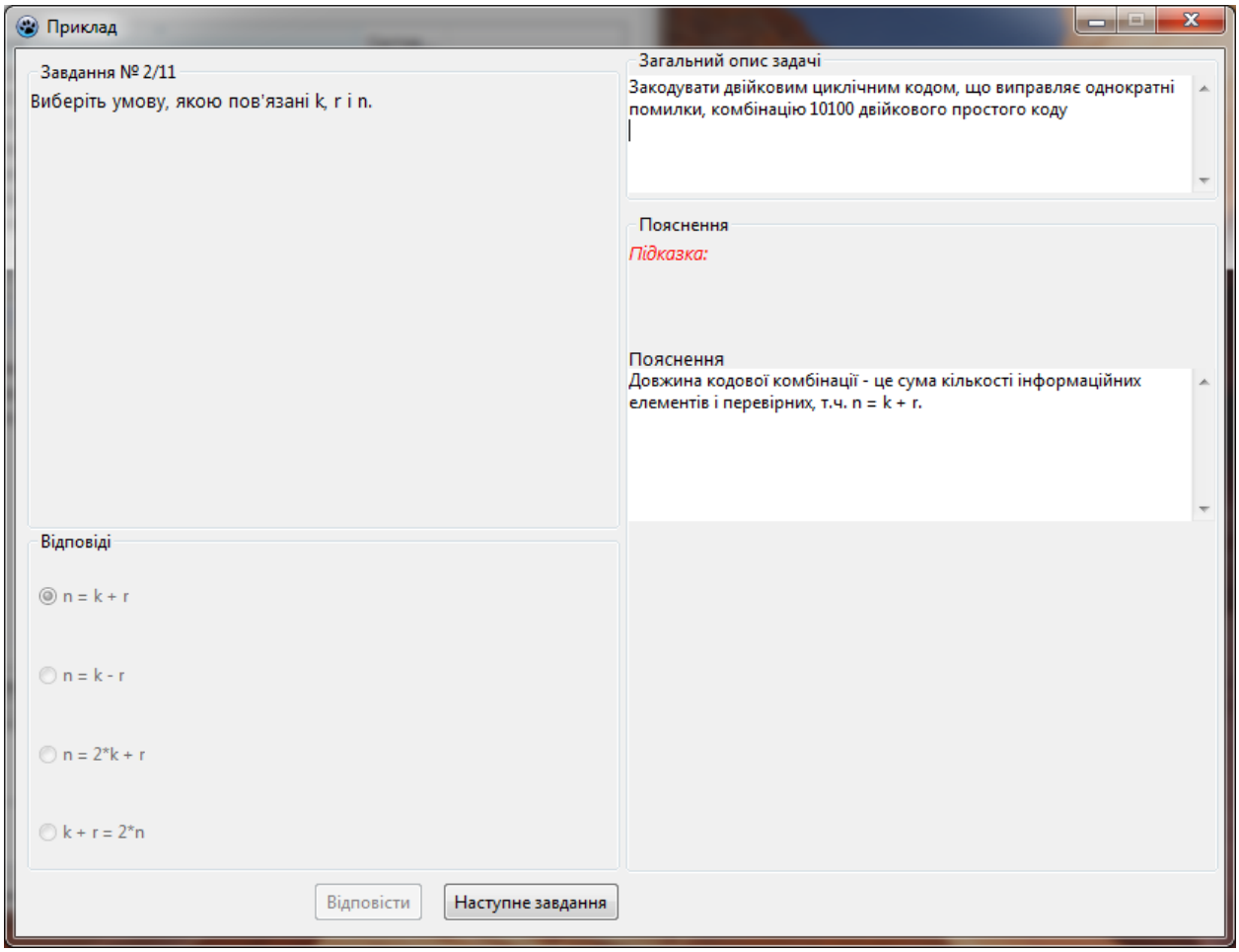

Рисунок 4.13 – Результати виконання другого завдання

Третє та четверте завдання є завданням з полем для відповіді. Згідно прикладу, якщо буде зроблено неправильну відповідь, то має з'явитись підказка, та буде надано можливість повторного введення відповіді. Якщо буде зроблено правильну відповідь, то буде виведно відповідне повідомлення. Результати виконання третього та четвертого завдань, зображені, відповідно, на рис. 4.14-4.15.

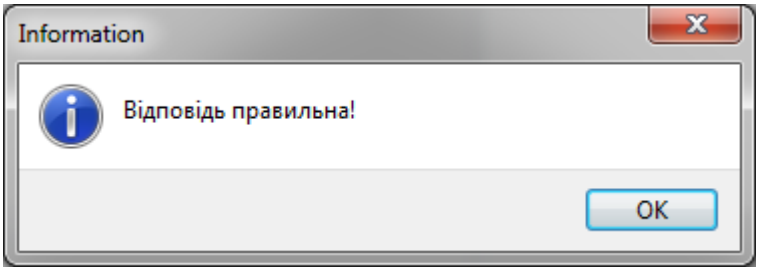

Рисунок 4.14 – Результати виконання третього завдання

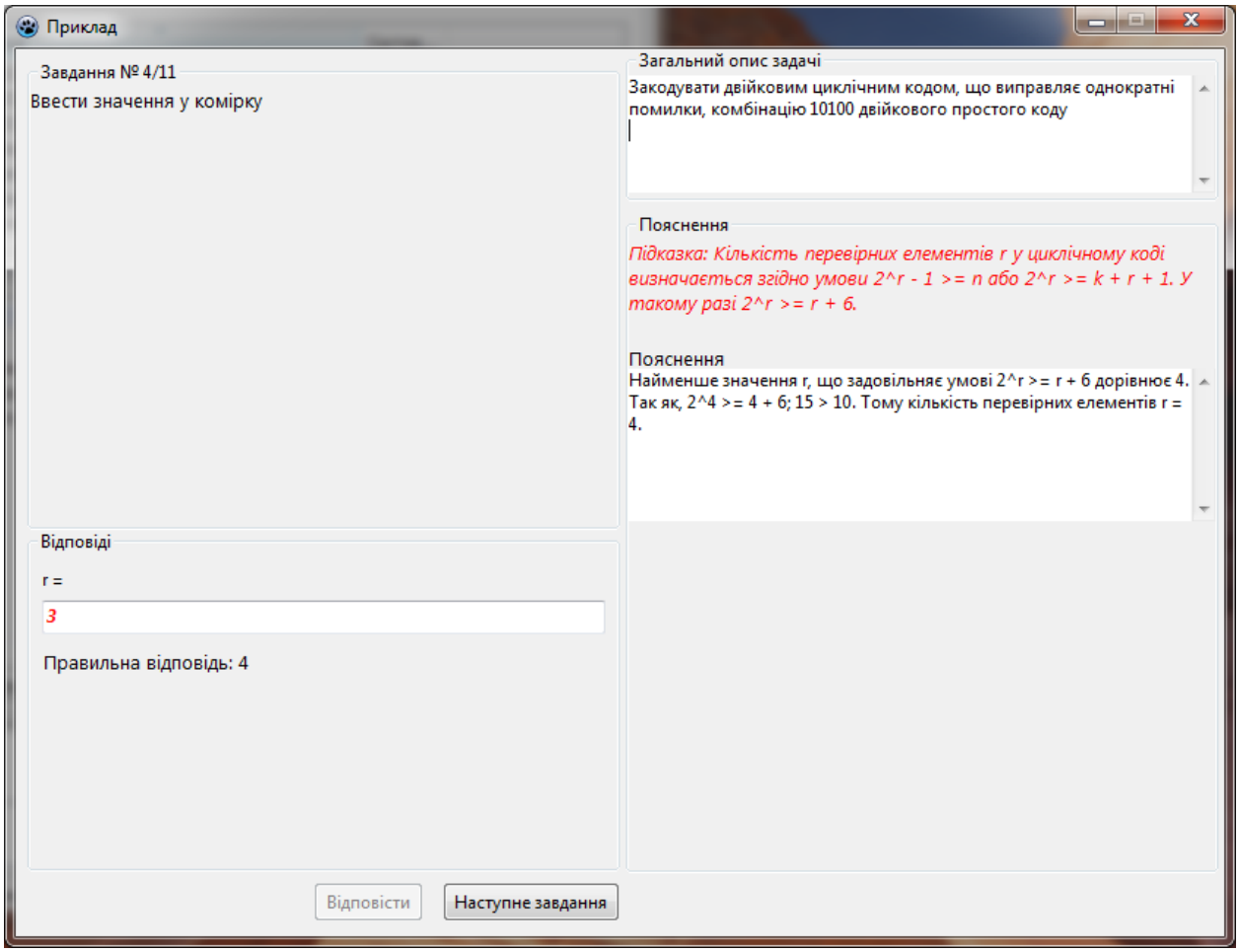

Рисунок 4.15 – Результати виконання четвертого завдання

П'яте завдання є завданням з декількома відповідями. До того ж в запитанні до завдання використовується допомоіжний матеріал у вигляді фотографії. Згідно прикладу, якщо буде зроблено неправильну відповідь, то має з'явитись підказка, та буде надано можливість повторного введення відповіді. Якщо буде зроблено правильну відповідь, то буде виведно відповідне повідомлення, інакше буде виведено повідомлення про помилку та пояснення до завдання.

Результати виконання п'ятого завдання, зображені на рис. 4.16.

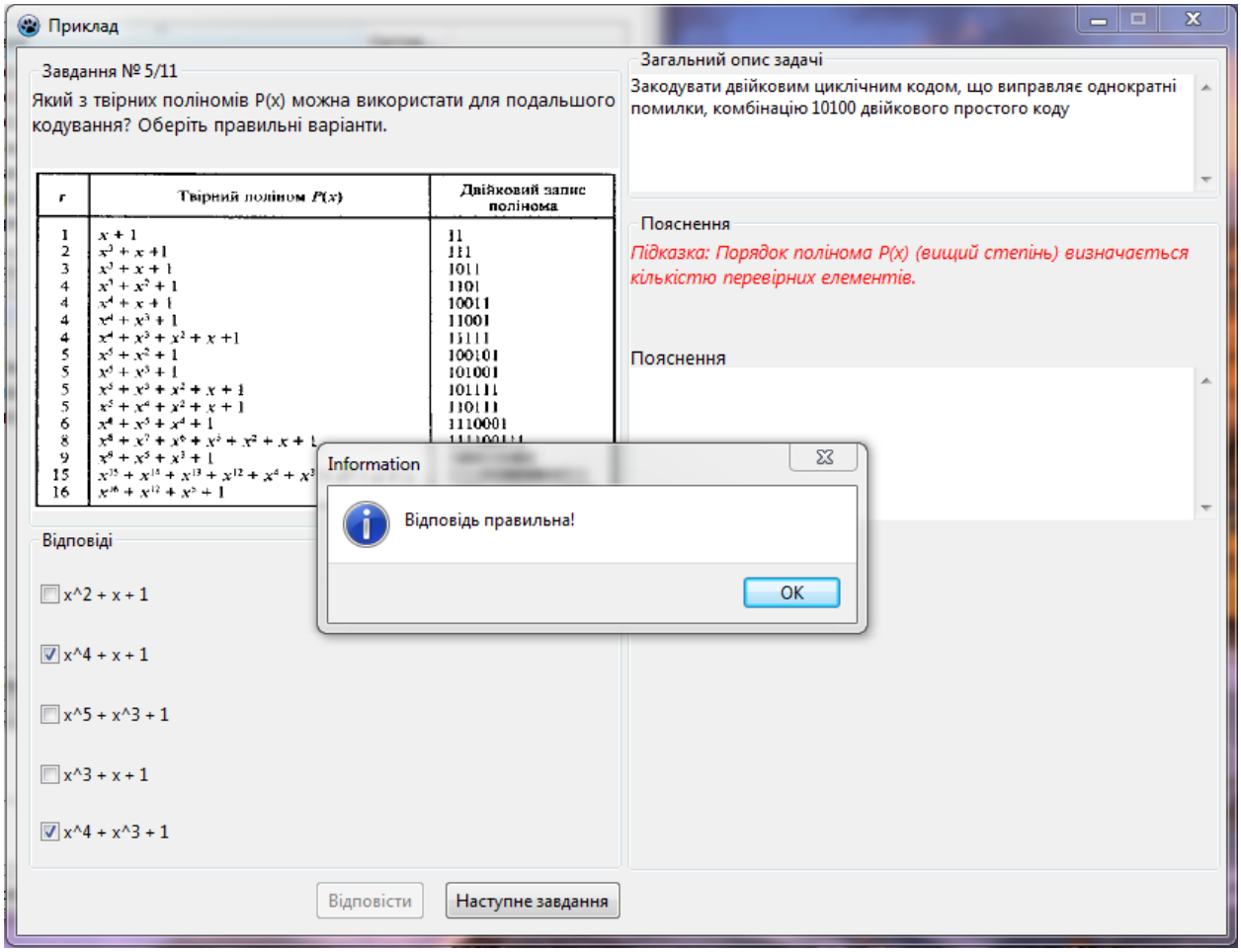

Рисунок 4.16 – Результати виконання п'ятого завдання

Завдання з номерами шість, сім та вісім є завданням з однією правильною відповідю та є ідентичними до розглянутого раніше завдання з номером два.

Завдання з номером дев'ять також є завданням з однією правильною відповідю, також в посненні до нього є допоміжний матеріал у вигляді фотографії. Згідно прикладу, якщо буде зроблено неправильну відповідь, то буде виведено повідомлення про помилку та пояснення до вирішення завдання. Результати виконання дев'ятого завдання, зображені на рис. 4.17.

Завдання з номерами десять та одинадцять є завданнями з однією правильною відповідю та є ідентичними до розглянутого раніше завдання з номером два.

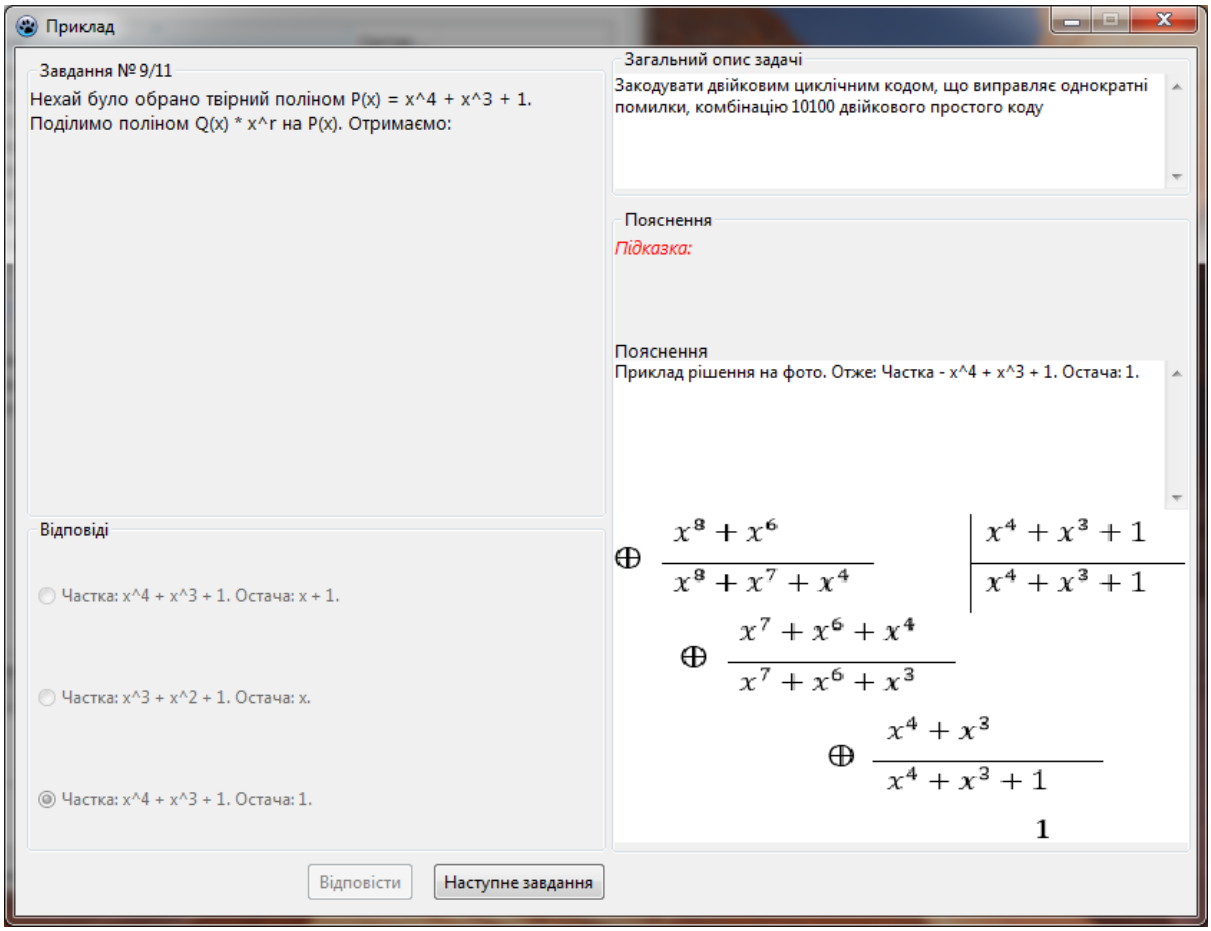

Рисунок 4.17 – Результати виконання дев'ятого завдання

Після виконання усіх завдань задачі, буде виведно повідомлення про результати виконання задачі, зображені на рис. 4.18.

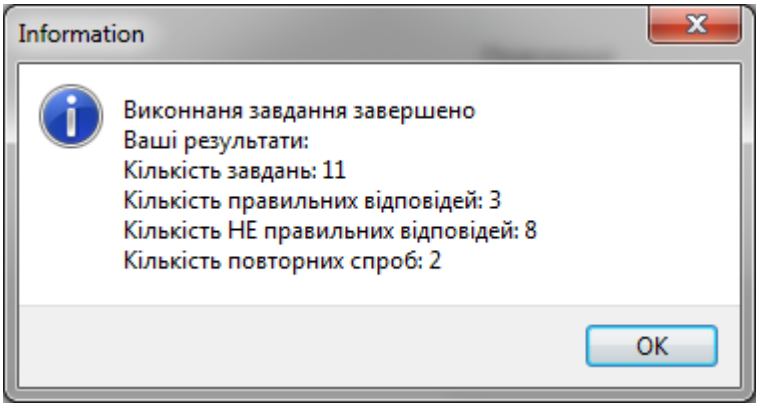

Рисунок 4.18 – Результати виконання задачі

В результаті тестування ПЗ можна зробити висновок, що всі функції тренажеру працюють справно та без помилок.

4.4 Інструкція користувача

Після запуску програми «eccsim.exe» перед користувачем з'являється логотип з привітанням, зображений на рис. 4.19, після чого з'являється вікно з обранням типу облікового запису, зображене на рис. 4.20.

Рисунок 4.19 – Логотип програми з привітанням

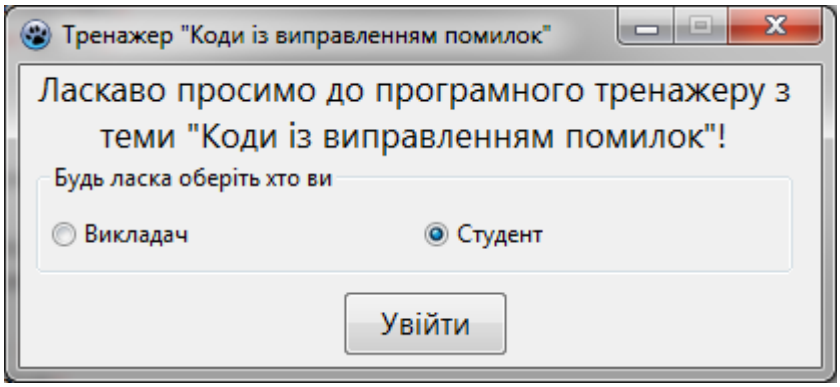

Рисунок 4.20 – Вікно обрання облікового запису

Якщо користувач обирає тип облікового запису «Студент», то він потрапляє до вікна, зображеному на рис. 4.21, з діями доступними «Студенту».

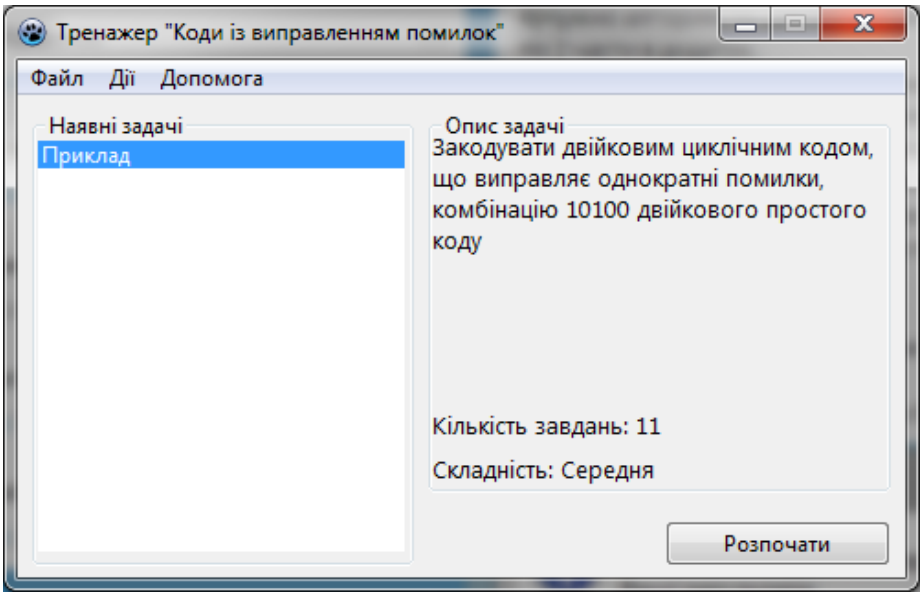

Рисунок 4.21 – Вікно з діями доступними «Студенту»

У цьому вікні користувач має можливість переглянути коротку характеристику вже існуючих задач та тестів використовуючи список задач у лівій частині вікна та короткий опис задач у правій. Також він може оновити список за допомогою меню «Дії» –> «Оновити список задач» або відкрити задачу, яка знаходиться в іншому місці комп'ютера за допомогою меню «Файл» –> «Відкрити задачу для виконання». Також користувач може переглянути інформацію про програму, зображену на рис. 4.22, прочитати інструкцію з її використання або прочитати теоретичні відомості за темою «Коди із виправлення помилок».

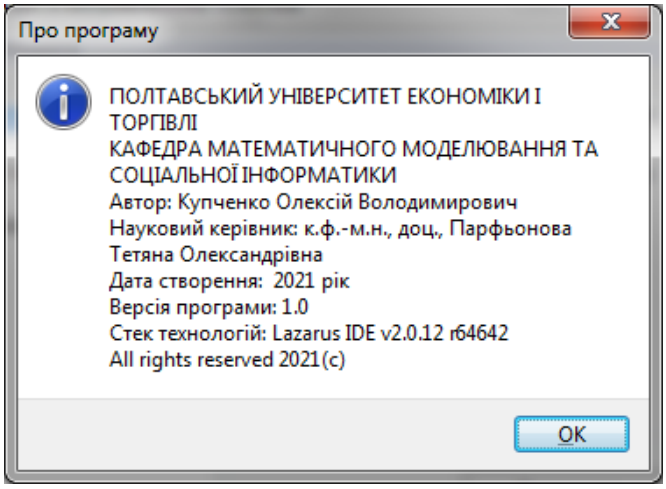

Рисунок 4.22 – Вікно «Про програму»

Вибравши задачу зі списку та натиснувши кнопку «Розпочати», або відкривши задачу з файлу, користувач потрапляє до вікна вирішення задач, зображеному на рис. 4.23.

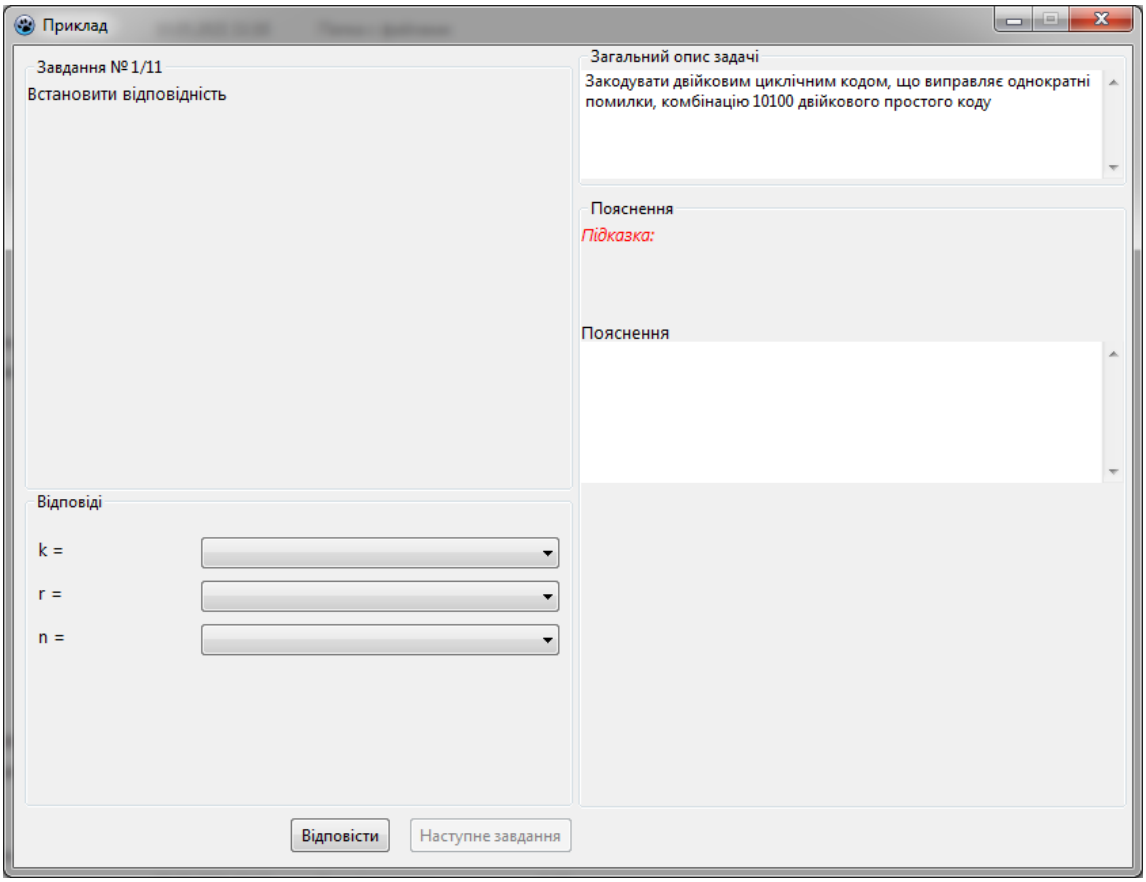

Рисунок 4.23 – Вікно «Вирішення задачі»

В цьому вікні в правому верхньому куті надається короткий опис задачі, зліва від розділ з завданням, який складається з опису запитання, його номер в задачі та фото якщо є. Під розділом з завданням знаходиться розділ відповідей, який отримає функціонал в залежності від типу завдання. Існують наступні тип завдань:

– завдання на співставлення відповідей, зображене на рис. 4.24;
– завдання з полем для відповіді, зображене на рис. 4.25;
– завдання з однією правильною відповіддю, зображене на рис. 4.26;
– завдання з декількома правильними відповідями, зображене на рис. 4.27.

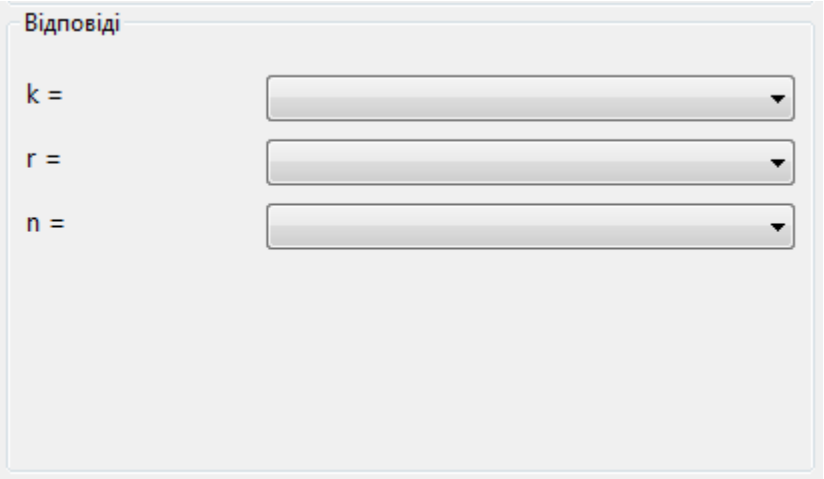

Рисунок 4.24 – Функціонал завдання на співставлення відповідей

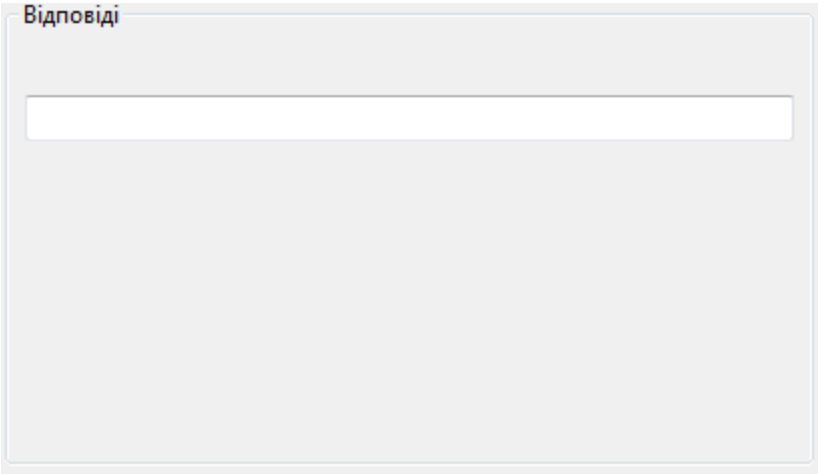

Рисунок 4.25 – Функціонал завдання з полем для відповіді

Рисунок 4.26 – Функціонал завдання з однією правильною відповіддю

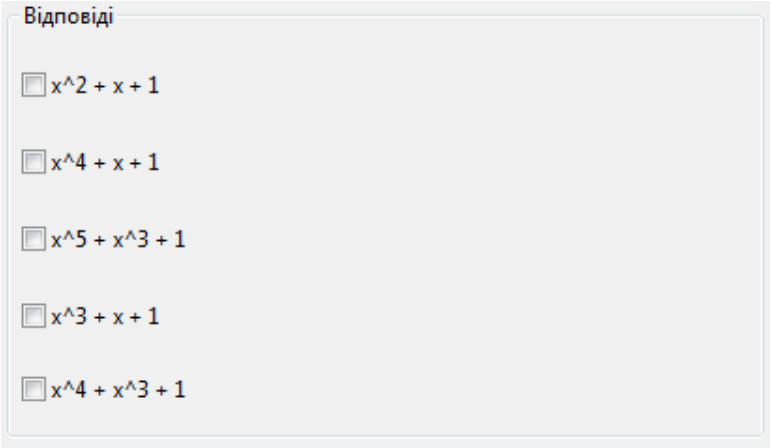

Рисунок 4.27 – Функціонал завдання з декількома правильними відповідями

Після обрання відповіді потрібно натиснути на кнопку «Відповісти». В правій частині вікна, під розділом з коротким описом програми, знаходиться розділ підказок та пояснень. Якщо користувач помилиться під час відповіді, то з'явиться відповідне повідомлення про помилку а також, якщо це передбачено завданням, то в розділі підказок та пояснень з'явиться підказка до завдання, та буде надано можливість повторного надання відповіді. Якщо ж користувач помилиться в другий раз, або підказки в завданні не передбачено, то автоматично буде надано потрібну відповідь та з'явиться пояснення до того як потрібно вирішити це завдання з можливим рисунок до пояснення, якщо звісно пояснення до завдання надається, а також можна бути перейти до наступного завдання за допомогою кнопки «Наступне завдання».

В кінці після надання всіх відповідей, буде виведено вікно з коротким описом результатів вирішення задачі, зображене на рис. 4.28.

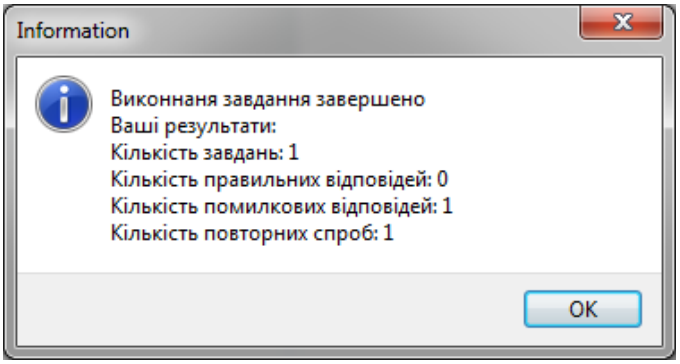

Рисунок 4.28 –Вікно з результатами вирішення задачі

Після цього користувач повернеться у вікно з вибором задач, де може вийти або вирішити іншу задачу. Якщо під час вибору облікового запису було обрано тип «Викладач», то з'явиться вікно вводу паролю для доступу до функціоналу, зображене на рис. 4.29.

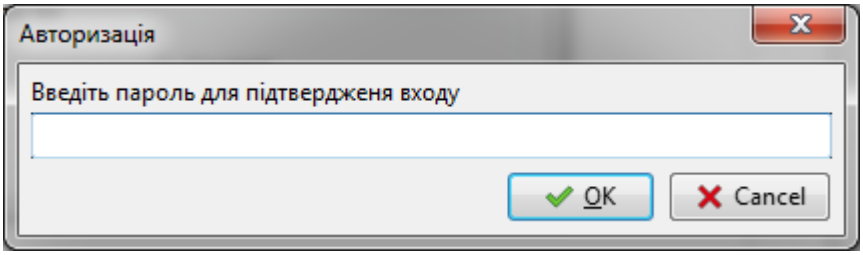

Рисунок 4.29 – Вікно вводу паролю

Після успішної авторизації користувач зможе створювати нові задачі, або редагувати існуючі, за допомогою відповідних кнопок у вікні дій, зображеному на рис. 4.30.

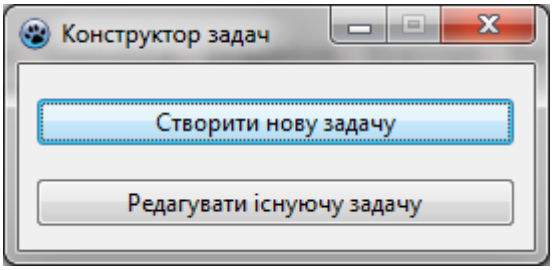

Рисунок 4.30 – Вікно дій користувача з обліковим записом «Викладач»

Після створення нової, або відкриття існуючої задачі, користувач потрапляє до головного вікна редагування задачі, зображеному на рис. 4.31.

В лівій частині вікна знаходиться список завдань наявних в задачі. В правій частині знаходяться розділи з загальним описом задачі, а саме назва та опис задачі, та розділ по роботі завданнями, де можна додавати нові задання, редагувати та видаляти існуючі.

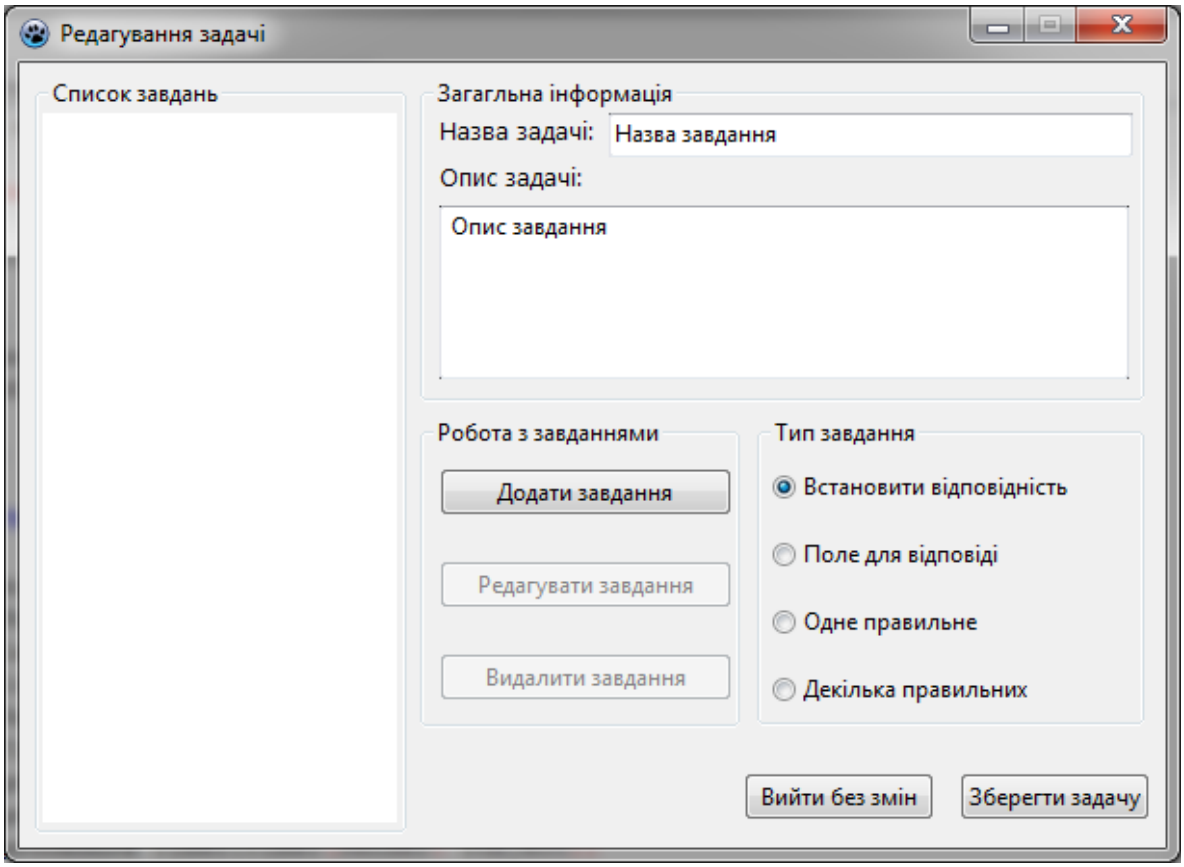

Рисунок 4.31 – Вікно редагування задачі

Для редагування та видалення завдань, потрібно обрати завдання зі списку та натиснути на відповідну кнопку. При додаванні нового завдання або редагування існуючого буде виведено вікно відповідно до типу завдання, зображене на рис.4.32-4.35.

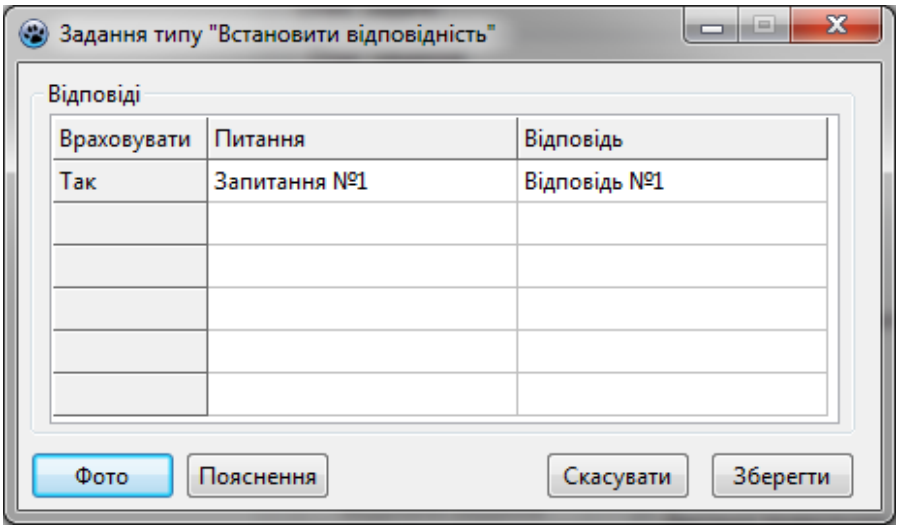

Рисунок 4.32 – Вікно редагування завдання на співставлення відповідей

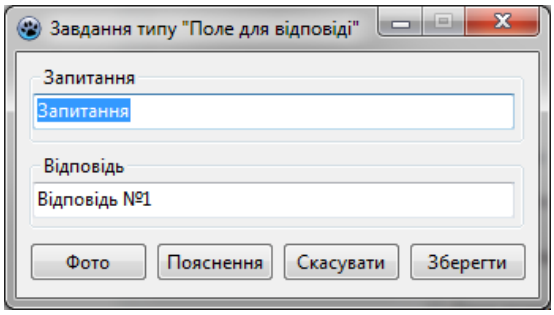

Рисунок 4.33 – Вікно редагування завдання з полем для відповіді

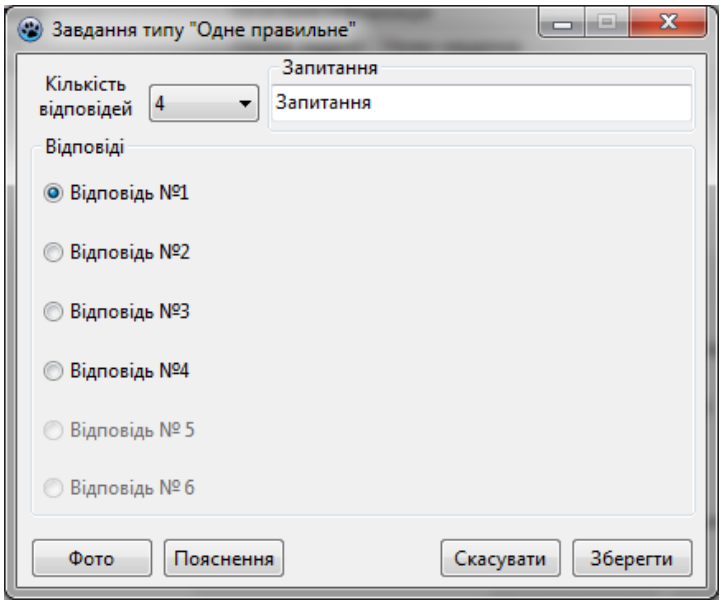

Рисунок 4.34 – Вікно редагування завдання з однією правильною відповіддю

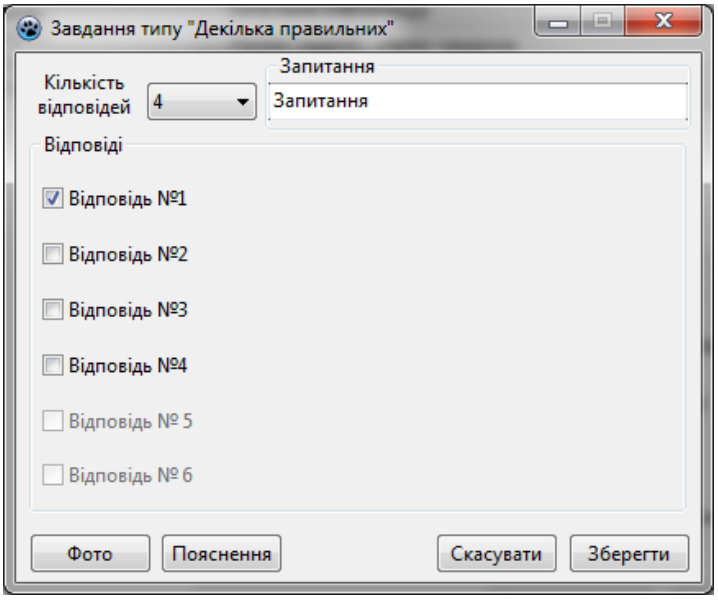

Рисунок 4.35 – Вікно редагування завдання з декількома правильними відповідями

В кожному з вікон можна редагувати запитання та відповіді, які є у завданні, а також: яка з відповідей є правильною, в тому випадку якщо відповідей декілька. Також до кожного завдання можна додати фото до запитання, та додати підказку і пояснення з фотографією, використовуючи інтерфейс відповідного вікна для створення та редагування підказок та пояснень до завдань, зображеного на рис. 4.36.

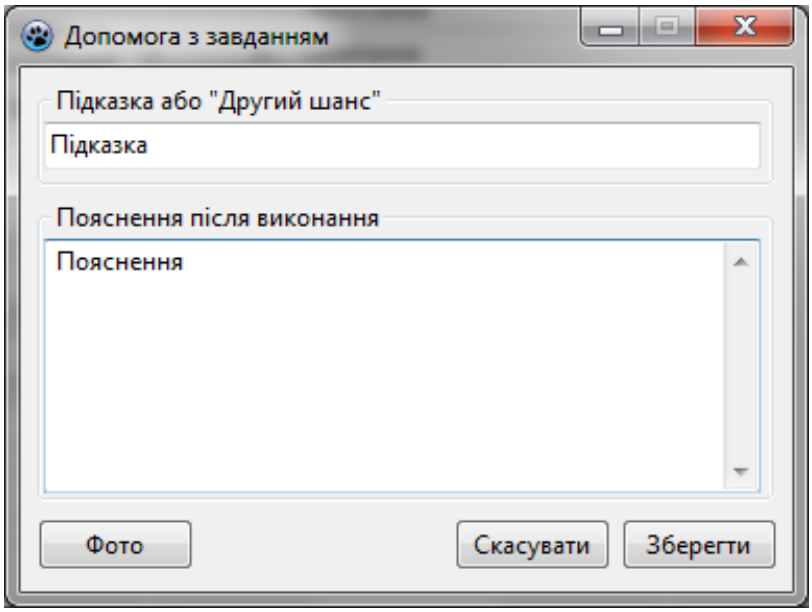

Рисунок 4.36 – Вікно додавання та редагування підказок та пояснень до завдання

ВИСНОВКИ

В ході роботи було виконано аналіз поставленого завдання, проведено інформаційний огляд існуючих засобів вирішення завдання, спроектовано, розроблено та проведено перевірку валідності програмного тренажеру, який дозволяє проводити навчання студентів з теми «Коди із виправленням помилок» дистанційного навчального курсу «Теорія інформації і кодування».

Спочатку було сформовано приклад для реалізації тренажеру та сформовані вимоги до тренажеру, потім було проведено інформаційний огляд існуючих аналогів, під час якого було проведено детальний аналіз та зроблені висновки щодо необхідності розробки тренажеру. Після чого була проведена алгоритмізація поставленої задачі, спираючись на теоретичні відомості про алгоритми з виправленням помилок, розроблено блок-схему алгоритму та обрані програмні засоби для розробки програмного додатку. В кінці було розроблено програмній додаток. Для цього були визначені склад графічного інтерфейсу користувача, спроектовані та розроблені програмні форми, для спрощення роботи користувачів з тренажером, проведене тестування роботи додатку на правильність роботи та розроблено інструкцію користувача.

Розроблений програмний застосунок дозволяє спростити та пришвидшити навчання студентів з теми Коди із виправленням помилок» дистанційного навчального курсу «Теорія інформації і кодування», а також полегшити роботу викладача та збільшити кількість студентів, які можуть одночасно вирішувати задачі навчального курсу з отриманням нових знань, не прибігаючи до допомоги викладача.

Підсумовуючи все вище сказане, можна зробити висновок, що в результаті виконання роботи була розроблена інформаційна система, яка в повному обсязі відповідає поставленим вимогам та виконує поставлені завдання.

СПИСОК ЛІТЕРАТУРИ

1. Аршинов М. Н., Садовский Л. Е. Коды и математика (рос.) / Михаил Аршинов Наумович, Леонид Ефимович Садовский // Москва: «Наука», 1983 — 145 с.

2. Блейхут Р. Э. Теория и практика кодов, контролирующих ошибки (рос.) — Вид. 1 — Москва: «Мир», 1986 — 576 с.

3. Жураковский Ю. П., Полторак В. П. Теорія інформації та кодування: Підручник / Ю. П. Жураковський, В. П. Полторак. // К.: Вища шк., 2001 — 255 с.

4. 12 платформ для онлайн образования [Електронний ресурс]. (Дата оновлення: 25.01.2021) Режим доступу (рос.): https://womo.ua/12-platform-dlya-onlayn-obrazovaniya/ (Дата звернення: 30.01.2021).

5. 13 образовательных онлайн-платформ с бесплатными курсами на английском и украинском [Електронний ресурс]. (Дата оновлення: 17.03.2020) Режим доступу (рос.): https://ain.ua/2020/03/17/obrazovatelnie-platformy/ (Дата звернення: 10.02.2021).

6. Coursera — Вікіпедія [Електронний ресурс]. (Дата оновлення: 06.02.2021) Режим доступу: https://uk.wikipedia.org/wiki/Coursera (Дата звернення: 16.02.2021).

7. Організація дистанційного навчання в Moodle — Освіта.UA [Електронний ресурс]. (Дата оновлення: 28.03.2020) Режим доступу: https://osvita.ua/vnz/high_school/72285/ (Дата звернення: 18.02.2021).

8. SoloLearn: Learn to Code [Електронний ресурс]. (Дата оновлення: 01.02.2021) Режим доступу (англ.): https://www.sololearn.com/home (Дата звернення: 23.02.2021).

9. Duolingo — Найкращий спосіб вивчати мову [Електронний ресурс]. (Дата оновлення: 14.01.2021) Режим доступу: https://uk.duolingo.com (Дата звернення: 23.02.2021).

10. Паскаль (язык программирования) — Википедия [Електронний ресурс]. (Дата оновлення: 12.05.2021) Режим доступу (рос.): https://ru.wikipedia.org/wiki/Паскаль_(язык_программирования)#Современные_версии_Object_Pascal (Дата звернення: 05.03.2021).

11. Lazarus — Вікіпедія [Електронний ресурс]. (Дата оновлення: 12.05.2021) Режим доступу: https://uk.wikipedia.org/wiki/Lazarus (Дата звернення: 05.03.2021).

12. Брукс Ф. Мифический человеко-месяц или как создаются программные системы: пер. з англ. / Ф. Брукс. (рос.) — Санкт-Петербург: Символ-Плюс, 1999. — 304 с.

13. Мирошниченко Е. А. Технологии программирования: учебное пособие / Е. А. Мирошниченко. (рос.) — Вид. 2, испр. и доп. — Томск: Изд-во Томского политехнического университета, 2008. — 128 с.

14. Итеративная и инкрементальная разработка: краткая история | Открытые системы. СУБД | Издательство «Открытые системы» [Електронний ресурс]. (Дата оновлення: 18.09.2003) Режим доступу: https://www.osp.ru/os/2003/09/183412 (Дата звернення: 11.04.2021).

15. McConnell S. Code Complete: A Practical Handbook of Software Construction (англ.) — Вид. 2. — Microsoft Press, 2004, — 960 с.

16. Ітеративна та інкрементна розробка — Вікіпедія [Електронний ресурс]. (Дата оновлення: 30.05.2020) Режим доступу: https://uk.wikipedia.org/wiki/ Ітеративна_та_інкрементна_розробка (Дата звернення: 25.04.2021).

ДОДАТОК А Текст програми

Файл exercise_unit.pas

```
unit exercise_unit;
{$mode objfpc}{$H+}
interface
uses  Classes, SysUtils, laz2_DOM, laz2_XMLRead, laz2_XMLWrite;
const
 TASK_TYPE_MATCH       = 1;  TASK_TYPE_FIELD       = 2;  TASK_TYPE_ONE_OF_MANY   = 3;
 TASK_TYPE_MANY_OF_MANY  = 4;  DEFAULT_ANSWER_COUNT    = 4;
 MAX_ANSWER_COUNT      = 6;  MAX_TASK_COUNT        = 20;
type
 AnswerRec    = Record
  LabelAnswer : String;    Text           : String;    IsRight    : Boolean;
 end;
        QuestionRec   = Record     Text       : String;  PhotoPath   : String;end;
 HelpRec      = Record    Hint       : String;    Solution    : String;    PhotoPath   : String;
 end;
        TaskRec      = Record
                TaskType            : Integer;
                Question            : QuestionRec;
                Answers             : Array [1..MAX_ANSWER_COUNT] of AnswerRec;
                AnswersCount: Integer;
                Help                 : HelpRec;
        end;
        ExerciseRec   = Record
                Title                : String;
                Description         : String;
                Difficult    : String;
                Tasks                : Array [1..MAX_TASK_COUNT] of TaskRec;
                TaskCount   : Integer;
        end;
 function LoadExerciseFromFile(
  FilePath : String;
  var Exercise : ExerciseRec) : boolean;
 function SaveExerciseToFile(
  FilePath : String;
  Exercise : ExerciseRec) : boolean;
```

```
implementation
function LoadExerciseFromFile(
  FilePath : String;
  var Exercise : ExerciseRec) : boolean;
var
  Doc : TXMLDocument;  Node, TaskNode, QNode, AnsNode, HelpNode : TDOMNode;  i, j : integer;
begin
  try
    ReadXMLFile(Doc, FilePath);
    Node := Doc.DocumentElement;
    if(Node.NodeName = 'ECC_SIM') then
    begin
      Node := Node.FirstChild;
      Exercise.Title := Node.TextContent;
      Node := Node.NextSibling;
      Exercise.Description := Node.TextContent;
      Node := Node.NextSibling;
      Exercise.Difficult := Node.TextContent;
      Node := Node.NextSibling;
      Exercise.TaskCount := Node.ChildNodes.Count;
      TaskNode := Node.FirstChild;
      i := 1;
      while Assigned(TaskNode) do
      begin
        with Exercise.Tasks[i] do
          begin
            TaskType := StrToInt(TaskNode.Attributes.Item[0].NodeValue);
            QNode            := TaskNode.ChildNodes.Item[0];
            Question.Text    := QNode.FirstChild.TextContent;
            Question.PhotoPath  := QNode.LastChild.TextContent;
            AnsNode      := TaskNode.ChildNodes.Item[1];
            AnswersCount  := AnsNode.ChildNodes.Count;
            AnsNode      := AnsNode.FirstChild;
            j := 1;
            while Assigned(AnsNode) do
            begin
              if(TaskType = TASK_TYPE_MATCH) or
                (TaskType = TASK_TYPE_FIELD) then
                  Answers[j].LabelAnswer := AnsNode.Attributes.Item[1].NodeValue;
```

```
          Answers[j].IsRight := StrToBool(AnsNode.Attributes.Item[0].NodeValue);
          Answers[j].Text := AnsNode.TextContent;
          j := j + 1;
          AnsNode := AnsNode.NextSibling;
        end;
        HelpNode     := TaskNode.ChildNodes.Item[2];
        Help.Hint    := HelpNode.ChildNodes.Item[0].TextContent;
        Help.Solution := HelpNode.ChildNodes.Item[1].TextContent;
        Help.PhotoPath:= HelpNode.ChildNodes.Item[2].TextContent;
       end;
      i := i + 1;
      TaskNode := TaskNode.NextSibling;
    end;
    Result := true;    end
   else     Result := false;
 except    Result := false;
 end;   Doc.Free;
end;
function SaveExerciseToFile(
 FilePath : String;
 Exercise : ExerciseRec) : boolean;
var
 Doc : TXMLDocument;
 RootNode, Node, TextNode, TaskNode, QNode, AnsNode, HelpNode : TDOMNode;
 i, j : integer;
begin
 try
   Doc := TXMLDocument.Create;
   RootNode := Doc.CreateElement('ECC_SIM');
   Doc.AppendChild(RootNode);
   RootNode  := Doc.DocumentElement;
   Node     := Doc.CreateElement('title');
   TextNode  := Doc.CreateTextNode(Exercise.Title);
   Node.AppendChild(TextNode);
   RootNode.AppendChild(Node);
   Node     := Doc.CreateElement('description');
   TextNode  := Doc.CreateTextNode(Exercise.Description);
   Node.AppendChild(TextNode);
   RootNode.AppendChild(Node);
```

```
Node     := Doc.CreateElement('difficult');
TextNode  := Doc.CreateTextNode(Exercise.Difficult);
Node.AppendChild(TextNode);
RootNode.AppendChild(Node);
Node     := Doc.CreateElement('tasks');
RootNode.AppendChild(Node);
RootNode  := Node;
for i := 1 to Exercise.TaskCount do
with Exercise.Tasks[i] do
begin
 TaskNode := Doc.CreateElement('task');
 TDOMElement(TaskNode).SetAttribute('attr', IntToStr(TaskType));
 case TaskType of
  TASK_TYPE_MATCH: TDOMElement(TaskNode).SetAttribute('name', 'MATCH');
  TASK_TYPE_FIELD: TDOMElement(TaskNode).SetAttribute('name', 'FIELD');
  TASK_TYPE_ONE_OF_MANY: TDOMElement(TaskNode).SetAttribute('name', 'ONE_OF_MANY');
  TASK_TYPE_MANY_OF_MANY: TDOMElement(TaskNode).SetAttribute('name', 'MANY_OF_MANY');
 end;
 QNode    := Doc.CreateElement('question');
 Node     := Doc.CreateElement('text');
 TextNode  := Doc.CreateTextNode(Question.Text);
 Node.AppendChild(TextNode);
 QNode.AppendChild(Node);
 Node     := Doc.CreateElement('photo');
 TextNode  := Doc.CreateTextNode(Question.PhotoPath);
 Node.AppendChild(TextNode);
 QNode.AppendChild(Node);
 TaskNode.AppendChild(QNode);
 AnsNode := Doc.CreateElement('answers');
 for j := 1 to AnswersCount do
 begin
   Node := Doc.CreateElement('answer');
   TDOMElement(Node).SetAttribute('attr', BoolToStr(Answers[j].IsRight));
   TDOMElement(Node).SetAttribute('label', Answers[j].LabelAnswer);
   TextNode := Doc.CreateTextNode(Answers[j].Text);
   Node.AppendChild(TextNode);
   AnsNode.AppendChild(Node);
 end;
 TaskNode.AppendChild(AnsNode);
```

```
    HelpNode  := Doc.CreateElement('help');
    Node      := Doc.CreateElement('hint');
    TextNode  := Doc.CreateTextNode(Help.Hint);
    Node.AppendChild(TextNode);
    HelpNode.AppendChild(Node);
    Node      := Doc.CreateElement('explanation');
    TextNode  := Doc.CreateTextNode(Help.Solution);
    Node.AppendChild(TextNode);
    HelpNode.AppendChild(Node);
    Node      := Doc.CreateElement('photo');
    TextNode  := Doc.CreateTextNode(Help.PhotoPath);
    Node.AppendChild(TextNode);
    HelpNode.AppendChild(Node);
    TaskNode.AppendChild(HelpNode);
    RootNode.AppendChild(TaskNode);
   end;
   writeXMLFile(Doc, FilePath);
   Result := true;
  except
   Result := false;
  end;
  Doc.Free;
end;
end.
```

Файл student_wnd.pas

```
unit student_main_wnd;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, Menus, ExtCtrls,  StdCtrls, FileUtil, ShellApi,
  exercise_creation_wnd, exercise_wnd, exercise_unit;
type
  TStudentMainWnd = class(TForm)
    btn_execute: TButton;    GroupBox1: TGroupBox;    GroupBox2: TGroupBox;
    description_label: TLabel;    MenuAction: TMenuItem;    MenuActionRefresh: TMenuItem;
    OpenDialog: TOpenDialog;    task_count_label: TLabel;    difficult_label: TLabel;
    ListBoxTests: TListBox;    MainMenu1: TMainMenu;    MenuItem1: TMenuItem;
    MenuFileOpen: TMenuItem;    MenuFileClose: TMenuItem;    MenuHelp: TMenuItem;
```

```
    MenuHelpAbout: TMenuItem;    MenuHelpHelp: TMenuItem;    MenuHelpInstruction: TMenuItem;
    procedure btn_executeClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ListBoxTestsClick(Sender: TObject);
    procedure MenuActionRefreshClick(Sender: TObject);
    procedure MenuFileCloseClick(Sender: TObject);
    procedure MenuFileOpenClick(Sender: TObject);
    procedure MenuHelpAboutClick(Sender: TObject);
    procedure MenuHelpHelpClick(Sender: TObject);
    procedure MenuHelpInstructionClick(Sender: TObject);
  private
  public
    ExerciseCount : Integer;    ExerciseList  : array [1..100] of ExerciseRec;
  end;
var
  StudentMainWnd: TStudentMainWnd;
implementation
{$R *.lfm}
{ TStudentMainWnd }
procedure TStudentMainWnd.MenuFileCloseClick(Sender: TObject);
begin  StudentMainWnd.Close;end;
procedure TStudentMainWnd.btn_executeClick(Sender: TObject);
begin
  if (ListBoxTests.ItemIndex <> -1) then
  begin
    ExerciseWnd.Exercise := ExerciseList[ListBoxTests.ItemIndex + 1];
    ExerciseWnd.ShowModal;
  end
  else
    MessageDlg('Помилка запуску',     'Не обрано жодної задачі для виконання!',  mtError,     [mbOk],  0);
end;
procedure TStudentMainWnd.FormCreate(Sender: TObject);
Var  files : TStringList;  i : integer;  Exercise : ExerciseRec;
begin
  files := FindAllFiles(ExtractFileDir(Paramstr(0)) + '\Exercises','*.eccsim', True);
  ExerciseCount := files.Count;
  i := 1;
  ExerciseCount := 0;
  for i := 0 to files.Count - 1 do
```

```
  if (LoadExerciseFromFile(files.Strings[i], Exercise)) then
  begin
    ExerciseCount := ExerciseCount + 1;
    ExerciseList[ExerciseCount] := Exercise;
    ListBoxTests.Items.Add(ExerciseList[ExerciseCount].Title);
  end;
 files.Free;
 if (ExerciseCount <> 0) then
 begin
  ListBoxTests.ItemIndex := 0;
  description_label.Caption := ExerciseList[1].Description;
  task_count_label.Caption := 'Кількість завдань: ' + IntToStr(ExerciseList[1].TaskCount);
  difficult_label.Caption := 'Складність: ' + ExerciseList[1].Difficult;
 end;
end;
procedure TStudentMainWnd.ListBoxTestsClick(Sender: TObject);
var  index : integer;
begin
 index := ListBoxTests.ItemIndex + 1;
 if (index <> 0) then
 begin
  description_label.Caption := ExerciseList[index].Description;
  task_count_label.Caption  := 'Кількість завдань: ' + IntToStr(ExerciseList[index].TaskCount);
  difficult_label.Caption   := 'Складність: ' + ExerciseList[index].Difficult;
 end;
end;
procedure TStudentMainWnd.MenuActionRefreshClick(Sender: TObject);
Var  files : TStringList;  i : integer;  Exercise : ExerciseRec;
begin
 ListBoxTests.Clear;
 files := FindAllFiles(ExtractFileDir(Paramstr(0)) + '\Exercises','*.eccsim', True);
 ExerciseCount := files.Count;
 i := 1;
 ExerciseCount := 0;
 for i := 0 to files.Count - 1 do
  if (LoadExerciseFromFile(files.Strings[i], Exercise)) then
  begin
    ExerciseCount := ExerciseCount + 1;
    ExerciseList[ExerciseCount] := Exercise;
```

```
    ListBoxTests.Items.Add(ExerciseList[ExerciseCount].Title);
   end;
  files.Free;
  if (ExerciseCount <> 0) then
  begin
   description_label.Caption := ExerciseList[1].Description;
   task_count_label.Caption := 'Кількість завдань: ' + IntToStr(ExerciseList[1].TaskCount);
   difficult_label.Caption := 'Складність: ' + ExerciseList[1].Difficult;
  end;
end;
procedure TStudentMainWnd.MenuFileOpenClick(Sender: TObject);
var  Exercise : ExerciseRec;
begin
  if (OpenDialog.Execute) then
   begin
    LoadExerciseFromFile(OpenDialog.FileName, Exercise);
    ExerciseWnd.Exercise := Exercise;
    ExerciseWnd.ShowModal;
   end;
end;
procedure TStudentMainWnd.MenuHelpAboutClick(Sender: TObject);
begin
  MessageDlg('Про програму',    'ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ' + #13#10 +
   'КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ' + #13#10 +
   'Автор: Купченко Олексій Володимирович' + #13#10 +
   'Науковий керівник: к.ф.-м.н., доц., Парфьонова Тетяна Олександрівна' + #13#10 +
   'Дата створення:  2021 рік' + #13#10 +    'Версія програми: 1.0' + #13#10 +
   'Стек технологій: Lazarus IDE v2.0.12 r64642'+ #13#10 +    'All rights reserved 2021(c)',
   mtInformation,    [mbOk],    0);
end;
procedure TStudentMainWnd.MenuHelpHelpClick(Sender: TObject);
begin  ShellExecute(0,'Open',Pchar(ExtractFileDir(Paramstr(0)) + '\theory.pdf'),nil,nil,1);end;
procedure TStudentMainWnd.MenuHelpInstructionClick(Sender: TObject);
begin  ShellExecute(0,'Open',Pchar(ExtractFileDir(Paramstr(0)) + '\instructions.pdf'),nil,nil,1);end;
end.
```

Файл exercise_wnd.pas

```
unit exercise_wnd;
{$mode objfpc}{$H+}
```

```
interface
uses  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ExtCtrls,  exercise_unit;
type
 TExerciseWnd = class(TForm)
   answer_label_1: TLabel;   answer_label_2: TLabel;   answer_label_4: TLabel;   answer_label_3: TLabel;
   answer_label_6: TLabel;   answer_label_5: TLabel;   answer_text_1: TComboBox;
   answer_text_2: TComboBox;   answer_text_3: TComboBox;   answer_text_4: TComboBox;
   answer_text_5: TComboBox;   answer_text_6: TComboBox;   btn_answer: TButton;   btn_next: TButton;
   gb_match:  TGroupBox;   gb_many_of_many:  TCheckGroup;       field_text_edit:  TEdit;  gb_exercise:
TGroupBox;   description_memo: TMemo;   gb_question: TGroupBox;   gb_help: TGroupBox;
   help_text_memo: TMemo;   gb_field: TGroupBox;   answer_label_field: TLabel;   help_photo: TImage;
   help_label: TLabel;   right_answer_label: TLabel;   question_photo: TImage;
   question_label: TLabel;   help_hint_label: TLabel;   gb_one_of_many: TRadioGroup;
   procedure btn_answerClick(Sender: TObject);
   procedure btn_nextClick(Sender: TObject);
   procedure FormCloseQuery(Sender: TObject; var CanClose: boolean);
   procedure FormCreate(Sender: TObject);
   procedure FormShow(Sender: TObject);
 private
   answers_labels : array[1..MAX_ANSWER_COUNT] of TLabel;
   answers_text : array[1..MAX_ANSWER_COUNT] of TComboBox;
   success_answers  : Integer;   second_chances   : Integer;   TaskNum : Integer;   SecondChance : boolean;
   procedure ShowTask();
   procedure ShowTaskMatch();
   procedure ShowTaskField();
   procedure ShowTaskOneOfMany();
   procedure ShowTaskManyOfMany();
   procedure ShowRightAnswers();
   function CheckAnswer() : boolean;
 public
   Exercise : ExerciseRec;
 end;
var
 ExerciseWnd: TExerciseWnd;
implementation
{$R *.lfm}
procedure TExerciseWnd.btn_answerClick(Sender: TObject);
var
 isok : boolean;  s : string;  p : integer;
```

```
begin
 isok := CheckAnswer();
 if (isok) or (not(isok) and not(SecondChance)) then
 begin
  btn_next.Enabled    := true;
  btn_answer.Enabled  := false;
 end;
 if not(isok) then
 begin
  MessageDlg(
   'Відповідь не правильна!',     mtError,     [mbOk],     0);
  if not(SecondChance) then
  begin
   s := Exercise.Tasks[TaskNum].Help.Solution;
   p := Pos(#10, s);
   while p <> 0 do
   begin
    if (s <> '') then
      help_text_memo.Lines.Add(Copy(s, 1, p));
    Delete(s, 1, p);
    p := Pos(#10, s);
   end;
   if (s <> '') then
      help_text_memo.Lines.Add(s);
   if (Exercise.Tasks[TaskNum].Help.PhotoPath <> 'null') then
    help_photo.Picture.LoadFromFile(Exercise.Tasks[TaskNum].Help.PhotoPath);
   ShowRightAnswers();
  end
  else
  begin
   help_hint_label.Caption :=
       'Підказка: ' + Exercise.Tasks[TaskNum].Help.Hint;
   SecondChance := false;
   btn_next.Enabled    := false;
   btn_answer.Enabled  := true;
  end;
 end
 else
  MessageDlg(     'Відповідь правильна!',     mtInformation,     [mbOk],     0);
```

```
  if(TaskNum = Exercise.TaskCount) then
     btn_next.Caption := 'Завершити';
 end;
 procedure TExerciseWnd.btn_nextClick(Sender: TObject);
 begin
  btn_next.Enabled       := false;
  btn_answer.Enabled     := true;
  help_hint_label.Caption := 'Підказка: ';
  help_text_memo.Clear;
  help_photo.Picture.Clear;
  if(TaskNum <> Exercise.TaskCount) then
  begin
   TaskNum := TaskNum + 1;
   ShowTask();
  end
  else
   begin
     MessageDlg(
      'Виконнаня завдання завершено' + #13#10 +       'Ваші результати: ' + #13#10 +
      'Кількість завдань: '       + IntToStr(Exercise.TaskCount) + #13#10 +
      'Кількість правильних відповідей: '
      + IntToStr(success_answers) + #13#10 +       'Кількість НЕ правильних відповідей: '
      + IntToStr(Exercise.TaskCount - success_answers) + #13#10 +       'Кількість повторних спроб: '
      + IntToStr(second_chances),       mtInformation,       [mbOk],       0);
     ExerciseWnd.Close;
  end;
 end;
 procedure TExerciseWnd.FormCloseQuery(Sender: TObject; var CanClose: boolean);
 var
  res : TModalResult;
 begin
  if(TaskNum <> Exercise.TaskCount) then
  begin
   res := MessageDlg('Завершення роботи','Ви ще не завершили роботу з задачею. Ви бажаєте
закінчити?',     mtConfirmation,     [mbYes, mbNo],     0);
   case res of
   mrYes:
   begin
    MessageDlg(
```

```
        'Виконнаня завдання завершено' + #13#10 +        'Ваші результати: ' + #13#10 +
        'Кількість завдань: '         + IntToStr(Exercise.TaskCount) + #13#10 +
        'Кількість правильних відповідей: '         + IntToStr(success_answers) + #13#10 +
        'Кількість НЕ правильних відповідей: ' + IntToStr(Exercise.TaskCount - success_answers) + #13#10 +
        'Кількість повторних спроб: '         + IntToStr(second_chances),        mtInformation,        [mbOk],   0);
     CanClose := true;
    end;
    mrNo: CanClose := false;
    end;
  end;
end;
procedure TExerciseWnd.FormCreate(Sender: TObject);
begin
  answers_labels[1] := answer_label_1;
  answers_labels[2] := answer_label_2;
  answers_labels[3] := answer_label_3;
  answers_labels[4] := answer_label_4;
  answers_labels[5] := answer_label_5;
  answers_labels[6] := answer_label_6;
  answers_text[1] := answer_text_1;
  answers_text[2] := answer_text_2;
  answers_text[3] := answer_text_3;
  answers_text[4] := answer_text_4;
  answers_text[5] := answer_text_5;
  answers_text[6] := answer_text_6;
end;
procedure TExerciseWnd.FormShow(Sender: TObject);
var
  s : string;  p : integer;
begin
  btn_next.Enabled       := false;
  btn_answer.Enabled     := true;
  btn_next.Caption       := 'Наступне завдання';
  ExerciseWnd.Caption := Exercise.Title;
  description_memo.Clear;
  s := Exercise.Description;
  p := Pos(#10, s);
  while p <> 0 do
  begin
```

```
  if (s <> '') then
    description_memo.Lines.Add(Copy(s, 1, p));
  Delete(s, 1, p);
  p := Pos(#10, s);
 end;
 if (s <> '') then
    description_memo.Lines.Add(s);
 success_answers := 0;
 second_chances  := 0;
 TaskNum := 1;
 ShowTask();
end;
procedure TExerciseWnd.ShowTaskMatch();
var
 i, j, k, n : integer;  arr : TStringList;
begin
 arr := TStringList.Create;
 with Exercise.Tasks[TaskNum] do
 begin
  for j := 1 to AnswersCount do
   arr.Add(Answers[j].Text);
  n := arr.Count;
  for i := 1 to MAX_ANSWER_COUNT do
  begin
   answers_text[i].Clear;
   answers_text[i].ItemIndex := -1;
   if (i <= AnswersCount) and
     (Answers[i].IsRight) then
   begin
    answers_labels[i].Caption := Answers[i].LabelAnswer;
    answers_labels[i].Font.Color  := clDefault;
    answers_labels[i].Font.Italic := false;
    answers_labels[i].Font.Bold   := false;
    answers_labels[i].Visible := True;
    Randomize;
    for j := 1 to N - 1 do
    begin
     k := Random(N - j + 1) + j;
     if k <> j then arr.Exchange(j - 1, k - 1);
```

```
      end;
      answers_text[i].Items.AddStrings(arr);
      answers_text[i].Enabled   := True;
      answers_text[i].Visible   := True;
     end
     else
     begin
      answers_labels[i].Visible := False;
      answers_text[i].Visible   := False;
     end;
    end;
    gb_match.Visible := True;
  end;
  arr.Destroy;
end;
procedure TExerciseWnd.ShowTaskField();
begin
  field_text_edit.Enabled     := True;
  field_text_edit.Font.Color  := clDefault;
  field_text_edit.Font.Italic := False;
  field_text_edit.Font.Bold   := False;
  field_text_edit.Text        := '';
  right_answer_label.Caption  := '';
  answer_label_field.Caption  := Exercise.Tasks[TaskNum].Answers[1].LabelAnswer;
  gb_field.Visible := True;
end;
procedure TExerciseWnd.ShowTaskOneOfMany();
var
  i : integer;
begin
  gb_one_of_many.Items.Clear;
  gb_one_of_many.ItemIndex := -1;
  gb_one_of_many.Visible := True;
  for i := 0 to Exercise.Tasks[TaskNum].AnswersCount - 1 do
  begin
    gb_one_of_many.Items.Add(Exercise.Tasks[TaskNum].Answers[i + 1].Text);
    gb_one_of_many.Controls[i].Font.Color   := clDefault;
    gb_one_of_many.Controls[i].Font.Italic  := false;
    gb_one_of_many.Controls[i].Font.Bold    := false;
```

```
  end;
end;
procedure TExerciseWnd.ShowTaskManyOfMany();
var
  i : integer;
begin
  gb_many_of_many.Items.Clear;
  gb_many_of_many.Visible := True;
  for i := 0 to Exercise.Tasks[TaskNum].AnswersCount - 1 do
  begin
    gb_many_of_many.Items.Add(Exercise.Tasks[TaskNum].Answers[i + 1].Text);
    gb_many_of_many.Controls[i].Font.Color   := clDefault;
    gb_many_of_many.Controls[i].Font.Italic  := false;
    gb_many_of_many.Controls[i].Font.Bold    := false;
  end;
end;
procedure TExerciseWnd.ShowTask();
begin
  gb_question.Caption  := 'Завдання № ' + IntToStr(TaskNum) +  '/' + IntToStr(Exercise.TaskCount);
  with Exercise.Tasks[TaskNum] do
  begin
    question_label.Caption  := Question.Text;
    if (Question.PhotoPath <> 'null') then
      question_photo.Picture.LoadFromFile(Question.PhotoPath)
    else
      question_photo.Picture.Clear;
    help_hint_label.Caption:= 'Підказка:';
    help_text_memo.Clear;
    if (Help.Hint <> '') then
      SecondChance := true
    else
    begin
      SecondChance := false;
      help_photo.Picture.Clear;
    end;
    gb_match.Visible       := False;
    gb_field.Visible       := False;
    gb_one_of_many.Visible  := False;
    gb_many_of_many.Visible := False;
```

```
   case (TaskType) of
   TASK_TYPE_MATCH:       ShowTaskMatch;
   TASK_TYPE_FIELD:       ShowTaskField;
   TASK_TYPE_ONE_OF_MANY:  ShowTaskOneOfMany;
   TASK_TYPE_MANY_OF_MANY: ShowTaskManyOfMany;
   end;
  end;
end;
procedure TExerciseWnd.ShowRightAnswers();
var
  i, j : Integer;
begin
  with Exercise.Tasks[TaskNum] do
    case (TaskType) of
      TASK_TYPE_MATCH:
      for i := 1 to AnswersCount do
      begin
        if (answers_text[i].ItemIndex <> -1) and
          (answers_text[i].Items[answers_text[i].ItemIndex] = Answers[i].Text)
          then
            answers_labels[i].Font.Color  := clGreen
        else
        begin
          answers_labels[i].Font.Color    := clRed;
          answers_labels[i].Font.Italic   := true;
          answers_labels[i].Font.Bold     := true;
          for j := 1 to answers_text[i].Items.Count do
            if (answers_labels[i].Caption = Answers[j].LabelAnswer) then
            begin
              answers_text[i].ItemIndex := j - 1;
              break;
            end;
        end;
        answers_text[i].Enabled := False;
      end;
      TASK_TYPE_FIELD:
      begin
        field_text_edit.Font.Color  := clRed;
        field_text_edit.Font.Italic := True;
```

```
      field_text_edit.Font.Bold   := True;
      right_answer_label.Caption  := 'Правильна відповідь: ' + Answers[1].Text;
    end;
    TASK_TYPE_ONE_OF_MANY:
    for i := 0 to AnswersCount - 1 do
    begin
     {if (gb_one_of_many.ItemIndex = i) and (Answers[i + 1].IsRight) then
       gb_one_of_many.Controls[i].Font.Color   := clRed;}
     if (Answers[i + 1].IsRight) then
     begin         gb_one_of_many.ItemIndex := i;       end;
    end;
    TASK_TYPE_MANY_OF_MANY:
    for i := 0 to AnswersCount - 1 do
    begin
     if (Answers[i + 1].IsRight) then
       gb_many_of_many.Checked[i] := true;
    end;
   end;
end;
function TExerciseWnd.CheckAnswer() : boolean;
var
 i: Integer;
begin
 Result := false;
 with Exercise.Tasks[TaskNum] do
  case (TaskType) of
   TASK_TYPE_MATCH:
   begin
    Result := true;
    for i := 1 to AnswersCount do
      if (Answers[i].IsRight) then
        if (answers_text[i].ItemIndex = -1) or
          (answers_text[i].Items[answers_text[i].ItemIndex] <>
           Answers[i].Text) then
        begin
          Result := false;
          break;
        end;
    end;
```

```
    TASK_TYPE_FIELD:
     if (field_text_edit.Text = Answers[1].Text) then
       Result := true;
    TASK_TYPE_ONE_OF_MANY:
     if (gb_one_of_many.ItemIndex <> -1) then
       if (Answers[gb_one_of_many.ItemIndex + 1].IsRight) then
         Result := true;
    TASK_TYPE_MANY_OF_MANY:
    begin
     Result := true;
     for i := 1 to AnswersCount do
       if (gb_many_of_many.Checked[i - 1] <> Answers[i].IsRight) then
       begin
         Result := false;
         break;
       end;
    end;
   end;
  if Result then    success_answers := success_answers + 1
  else if SecondChance then    second_chances := second_chances + 1;
end;
end.
```

Файл main_wnd.pas

```
unit main_wnd;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, EditBtn,
  StdCtrls, Windows, student_main_wnd, teacher_main_wnd;
type
  TMainWnd = class(TForm)
    Button1: TButton;    Label1: TLabel;    logo: TImage;    RadioGroup1: TRadioGroup;    Timer1: TTimer;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
  public
  end;
```

```
var
  MainWnd: TMainWnd;
implementation
{$R *.lfm}
procedure TMainWnd.FormCreate(Sender: TObject);
begin
  RadioGroup1.ItemIndex := 1;
end;
procedure TMainWnd.Button1Click(Sender: TObject);
var
  value : string;
begin
  if RadioGroup1.ItemIndex = 0 then
  begin
    value := InputBox('Авторизація',            'Введіть Пароль для підтвердженя входу','');
    if (value <> 'teacher') then
      MessageDlg('Помилка авторизації',     'Неправильний пароль!',     mtError,     [mbOk],     0)
    else
    begin
      MainWnd.Visible := False;
      TeacherMainWnd.ShowModal;
      MainWnd.Visible := True;
    end;
  end
  else
  begin
    MainWnd.Visible := False;
    StudentMainWnd.ShowModal;
    MainWnd.Visible := True;
  end;
end;
procedure TMainWnd.Timer1Timer(Sender: TObject);
begin
  Timer1.Enabled := False;
  logo.Visible := False;

  RadioGroup1.Visible := True;
  Label1.Visible := True;
  Button1.Visible := True;
```

```
  MainWnd.BorderStyle := bsSingle;
  MainWnd.Height      := 150;
  MainWnd.Width       := 380;
  MainWnd.Position    := poDesktopCenter;
end;
end.
```

Файл teacher_main_wnd.pas

```
unit teacher_main_wnd;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,  exercise_creation_wnd, exercise_unit;
type
  TTeacherMainWnd = class(TForm)
    btn_new: TButton;    btn_edit: TButton;    OpenDialog: TOpenDialog;
    procedure btn_editClick(Sender: TObject);
    procedure btn_newClick(Sender: TObject);
  private
  public
  end;
var
  TeacherMainWnd: TTeacherMainWnd;
implementation
{$R *.lfm}
procedure TTeacherMainWnd.btn_newClick(Sender: TObject);
var
  Exercise : ExerciseRec;
begin
  Exercise.Title       := 'Назва завдання';
  Exercise.Description  := 'Опис завдання';
  Exercise.Difficult    := 'Складність завдання';
  Exercise.TaskCount    := 0;
  ExerciseCreationWnd.Exercise := Exercise;
  ExerciseCreationWnd.ShowModal;
end;
procedure TTeacherMainWnd.btn_editClick(Sender: TObject);
var
  Exercise : ExerciseRec;
```

```
begin
 if (OpenDialog.Execute) then
  begin
   LoadExerciseFromFile(OpenDialog.FileName, Exercise);
   ExerciseCreationWnd.Exercise := Exercise;
   ExerciseCreationWnd.ShowModal;
  end;
end;
end.
```

Файл exercise_creation_wnd.pas

```
unit exercise_creation_wnd;
{$mode objfpc}{$H+}
interface
uses
 Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, ExtCtrls,
 task_field_wnd, task_oneofmany_wnd, task_manyofmany_wnd, task_match_wnd,
 exercise_unit;
const
 DEFAULT_ANSWER_COUNT = 4;
type
 TExerciseCreationWnd = class(TForm)
  btn_edit_task: TButton;  btn_add_task: TButton;  btn_save_exercise: TButton;  btn_delete_task:
TButton;  btn_exit_no_changes: TButton;  GroupBox3: TGroupBox;
  Label2: TLabel;  description_memo: TMemo;  rg_tasktype: TRadioGroup;  SaveDialog: TSaveDialog;
  title_edit: TEdit;  GroupBox1: TGroupBox;  GroupBox2: TGroupBox;  Label1: TLabel;  TaskList: TListBox;
  procedure btn_add_taskClick(Sender: TObject);
  procedure btn_delete_taskClick(Sender: TObject);
  procedure btn_edit_taskClick(Sender: TObject);
  procedure btn_exit_no_changesClick(Sender: TObject);
  procedure btn_save_exerciseClick(Sender: TObject);
  procedure description_memoChange(Sender: TObject);
  procedure FormCloseQuery(Sender: TObject; var CanClose: boolean);
  procedure FormShow(Sender: TObject);
  procedure rg_tasktypeSelectionChanged(Sender: TObject);
  procedure TaskListClick(Sender: TObject);
  procedure title_editChange(Sender: TObject);
 private
  NeedToSave : boolean;
```

```
  public
    Exercise : ExerciseRec;
  end;
var
  ExerciseCreationWnd: TExerciseCreationWnd;
implementation
{$R *.lfm}
procedure TExerciseCreationWnd.TaskListClick(Sender: TObject);
var
  index : integer;
begin
  index := TaskList.ItemIndex + 1;
  if(index <> 0) then
  begin
    rg_tasktype.ItemIndex   := Exercise.Tasks[index].TaskType - 1;
    btn_edit_task.Enabled   := true;
    btn_delete_task.Enabled := true;
  end;
end;
procedure TExerciseCreationWnd.title_editChange(Sender: TObject);
begin  NeedToSave := true;end;
procedure TExerciseCreationWnd.btn_add_taskClick(Sender: TObject);
var
  Task : TaskRec;
  i : integer;
begin
  if (Exercise.TaskCount <> MAX_TASK_COUNT) then
  begin
    Task.Question.PhotoPath := 'null';
    Task.Question.Text      := 'Запитання';
    Task.Help.Hint          := 'Підказка';
    Task.Help.Solution      := 'Пояснення';
    Task.Help.PhotoPath     := 'null';
    Task.AnswersCount       := DEFAULT_ANSWER_COUNT;
    for i := 1 to MAX_ANSWER_COUNT do
    begin
      Task.Answers[i].Text       := 'Відповідь №' + IntToStr(i);
      Task.Answers[i].LabelAnswer := 'Запитання №' + IntToStr(i);
      Task.Answers[i].IsRight     := false;
```

```
end;
Task.Answers[1].IsRight := True;
case (rg_tasktype.ItemIndex) of
 0:
 begin
  Task.AnswersCount := 1;
  Task.TaskType     := TASK_TYPE_MATCH;
  TaskMatchWnd.Task := Task;
  TaskMatchWnd.ShowModal;
  Task := TaskMatchWnd.Task;
 end;
 1:
 begin
  Task.TaskType     := TASK_TYPE_FIELD;
  TaskFieldWnd.Task := Task;
  TaskFieldWnd.ShowModal;
  Task := TaskFieldWnd.Task;
 end;
 2:
 begin
  Task.TaskType        := TASK_TYPE_ONE_OF_MANY;
  TaskOneOfManyWnd.Task := Task;
  TaskOneOfManyWnd.ShowModal;
  Task := TaskOneOfManyWnd.Task;
 end;
 3:
 begin
  Task.TaskType          := TASK_TYPE_MANY_OF_MANY;
  TaskManyOfManyWnd.Task  := Task;
  TaskManyOfManyWnd.ShowModal;
  Task := TaskManyOfManyWnd.Task;
 end;
end;
Exercise.TaskCount := Exercise.TaskCount + 1;
Exercise.Tasks[Exercise.TaskCount] := Task;
TaskList.Items.Add('Завдання № ' + IntToStr(Exercise.TaskCount));
TaskList.ItemIndex     := -1;
btn_edit_task.Enabled   := false;
btn_delete_task.Enabled := false;
```

```
    NeedToSave            := true;
  end
  else
    MessageDlg('Додавання даних',     'Помилка при додавані даних!' + #13#10 +
      'Максимальна кількість завдань: ' + IntToStr(MAX_ANSWER_COUNT),     mtError,     [mbOk],     0);
end;
procedure TExerciseCreationWnd.btn_delete_taskClick(Sender: TObject);
var
  i, j, del : integer;
begin
  del := TaskList.ItemIndex;
  if (del <> -1) then
  begin
    TaskList.Clear;
    Exercise.TaskCount := Exercise.TaskCount - 1;
    for i := 0 to Exercise.TaskCount - 1 do
      TaskList.Items.Add('Завдання № ' + IntToStr(i + 1));
    for i := del + 1 to Exercise.TaskCount do
    begin
      Exercise.Tasks[i].TaskType     := Exercise.Tasks[i + 1].TaskType;
      Exercise.Tasks[i].Help         := Exercise.Tasks[i + 1].Help;
      Exercise.Tasks[i].Question     := Exercise.Tasks[i + 1].Question;
      Exercise.Tasks[i].AnswersCount  := Exercise.Tasks[i + 1].AnswersCount;
      for j := 1 to Exercise.Tasks[i].AnswersCount do
        Exercise.Tasks[i].Answers[j]  := Exercise.Tasks[i + 1].Answers[j];
    end;
    TaskList.ItemIndex     := -1;
    btn_edit_task.Enabled   := false;
    btn_delete_task.Enabled := false;
    NeedToSave            := true;
  end;
end;
procedure TExerciseCreationWnd.btn_edit_taskClick(Sender: TObject);
var
  index : integer;
begin
  index := TaskList.ItemIndex + 1;
  if(index <> 0) then
    case Exercise.Tasks[index].TaskType of
```

```
    TASK_TYPE_MATCH:
    begin
     TaskMatchWnd.Task := Exercise.Tasks[index];
     TaskMatchWnd.ShowModal;
     Exercise.Tasks[index] := TaskMatchWnd.Task;
    end;
    TASK_TYPE_FIELD:
    begin
     TaskFieldWnd.Task := Exercise.Tasks[index];
     TaskFieldWnd.ShowModal;
     Exercise.Tasks[index] := TaskFieldWnd.Task;
    end;
    TASK_TYPE_ONE_OF_MANY:
    begin
     TaskOneOfManyWnd.Task := Exercise.Tasks[index];
     TaskOneOfManyWnd.ShowModal;
     Exercise.Tasks[index] := TaskOneOfManyWnd.Task;
    end;
    TASK_TYPE_MANY_OF_MANY:
    begin
     TaskManyOfManyWnd.Task := Exercise.Tasks[index];
     TaskManyOfManyWnd.ShowModal;
     Exercise.Tasks[index] := TaskManyOfManyWnd.Task;
    end;
   end;
  TaskList.ItemIndex     := -1;
  btn_edit_task.Enabled   := false;
  btn_delete_task.Enabled := false;
  NeedToSave            := true;
end;
procedure TExerciseCreationWnd.btn_exit_no_changesClick(Sender: TObject);
begin
  NeedToSave := false;
  ExerciseCreationWnd.Close;
end;
procedure TExerciseCreationWnd.btn_save_exerciseClick(Sender: TObject);
begin
  if(Exercise.TaskCount = 0) then
    MessageDlg('Збереження даних',
```

```
    'Помилка при збереженні даних!' + #13#10 +     'Потрібно хоч одне завдання у списку завдань!',
    mtError,    [mbOk],    0)
  else if(SaveDialog.Execute) then
  begin
   Exercise.Title       := title_edit.Text;
   Exercise.Description  := description_memo.Text;
   Exercise.Difficult    := 'Середня';
   if (SaveExerciseToFile(SaveDialog.FileName, Exercise)) then
    MessageDlg('Збереження даних','Дані було успішно збережено!',    mtInformation,      [mbOk],   0)
   else
    MessageDlg('Збереження даних',    'Помилка при збереженні даних!' + #13#10 +
    'Спробуйте знов. Якщо це не допоможе, то зверніться до з описом проблеми!',
    mtError,    [mbOk],    0);
   NeedToSave := false;
  end;
end;
procedure TExerciseCreationWnd.description_memoChange(Sender: TObject);
begin  NeedToSave := true;end;
procedure TExerciseCreationWnd.FormCloseQuery(Sender: TObject;
  var CanClose: boolean);
var
  res : TModalResult;
begin
  if(NeedToSave) then
  begin
   res := MessageDlg('Задача змінена',    'Дані в задачі були змінені. Ви бажаєте зберегти?',
    mtConfirmation,    [mbYes, mbNo, mbCancel],    0);
   case res of
   mrYes:
   if(Exercise.TaskCount = 0) then
    MessageDlg('Збереження даних',       'Помилка при збереженні даних!' + #13#10 +
     'Потрібно хоч одне завдання у списку завдань!',       mtError,       [mbOk],       0)
   else
   begin
    btn_save_exerciseClick(Sender);
    CanClose := true;
   end;
   mrNo:    CanClose := true;
   mrCancel: CanClose := false;
```

```
    end;
  end;
end;
procedure TExerciseCreationWnd.FormShow(Sender: TObject);
var
  s : string;
  i : integer;
begin
  title_edit.Text   := Exercise.Title;
  description_memo.clear;
  s := Exercise.Description;
  i := Pos(#10, s);
  while i <> 0 do
  begin
    if (s <> '') then
      description_memo.Lines.Add(Copy(s, 1, i));
    Delete(s, 1, i);
    i := Pos(#10, s);
  end;
  if (s <> '') then
      description_memo.Lines.Add(s);
  TaskList.Clear;
  for i := 1 to Exercise.TaskCount do
    TaskList.Items.Add('Завдання № ' + IntToStr(i));
  NeedToSave := false;
end;
procedure TExerciseCreationWnd.rg_tasktypeSelectionChanged(Sender: TObject);
var
  index : integer;
begin
  index := TaskList.ItemIndex + 1;
  if(index <> 0) then
    Exercise.Tasks[index].TaskType := rg_TaskType.ItemIndex + 1;
end;
end.
```

Файл task_field_wnd.pas

```
unit task_field_wnd;
{$mode objfpc}{$H+}
```

```
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls,  explanation_wnd, exercise_unit;
type
  TTaskFieldWnd = class(TForm)
    btn_save: TButton;    btn_cancel: TButton;    btn_photo: TButton;    btn_explanation: TButton;
    Edit_answer: TEdit;    Edit_question: TEdit;    GroupBox1: TGroupBox;    GroupBox2: TGroupBox;
    PhotoOpen: TOpenDialog;
    procedure btn_cancelClick(Sender: TObject);
    procedure btn_explanationClick(Sender: TObject);
    procedure btn_photoClick(Sender: TObject);
    procedure btn_saveClick(Sender: TObject);
    procedure Edit_answerChange(Sender: TObject);
    procedure Edit_questionChange(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: boolean);
    procedure FormShow(Sender: TObject);
  private
    Help : HelpRec;    PhotoPath : String;    NeedToSave : boolean;
  public
    Task : TaskRec;
  end;
var
  TaskFieldWnd: TTaskFieldWnd;
implementation
{$R *.lfm}
procedure TTaskFieldWnd.btn_cancelClick(Sender: TObject);
begin
  NeedToSave := False;
  TaskFieldWnd.Close;
end;
procedure TTaskFieldWnd.btn_explanationClick(Sender: TObject);
begin
  ExplanationWnd.Help := Help;
  ExplanationWnd.ShowModal;
  Help       := ExplanationWnd.Help;
  NeedToSave  := True;
end;
procedure TTaskFieldWnd.btn_photoClick(Sender: TObject);
begin
```

```
 if (PhotoPath <> 'null') then
 begin
  if( MessageDlg('Видалення фото',
    'До завдання вже прив''язане фото. Ви бажаєте його видалити?',     mtConfirmation,
    [mbYes, mbNo],     0) = mrYes) then
      PhotoPath := 'null';
 end
 else if (PhotoOpen.Execute) then
  PhotoPath := PhotoOpen.FileName;
 NeedToSave := True;
end;
procedure TTaskFieldWnd.btn_saveClick(Sender: TObject);
begin
  Task.Question.Text      := Edit_question.Text;
  Task.Question.PhotoPath  := PhotoPath;
  Task.Answers[1].Text    := Edit_answer.Text;
  Task.Help              := Help;
  NeedToSave             := False;
  TaskFieldWnd.Close;
end;
procedure TTaskFieldWnd.Edit_answerChange(Sender: TObject);
begin  NeedToSave := True;end;
procedure TTaskFieldWnd.Edit_questionChange(Sender: TObject);
begin  NeedToSave := True;end;
procedure TTaskFieldWnd.FormCloseQuery(Sender: TObject; var CanClose: boolean);
var
 res : TModalResult;
begin
 if(NeedToSave) then
 begin
  res := MessageDlg('Збереження даних',     'Дані в завданні були змінені. Ви бажаєте зберегти?',
    mtConfirmation,     [mbYes, mbNo, mbCancel],     0);
  case res of
  mrYes:
  begin
    btn_saveClick(Self);
    CanClose := true;
  end;
  mrNo:    CanClose := true;
```

```
   mrCancel: CanClose := false;
   end;
  end;
end;
procedure TTaskFieldWnd.FormShow(Sender: TObject);
begin
  Edit_question.Text  := Task.Question.Text;
  PhotoPath          := Task.Question.PhotoPath;
  Edit_answer.Text    := Task.Answers[1].Text;
  Help               := Task.Help;
  NeedToSave         := False;
end;
end.
```

Файл task_manyofmany_wnd.pas

```
unit task_manyofmany_wnd;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, StdCtrls,
  Menus, explanation_wnd, exercise_unit;
type
  TTaskManyOfManyWnd = class(TForm)
    AnswersPopupMenu: TPopupMenu;    answer_count: TComboBox;    answer_edit: TMenuItem;
    btn_explanation: TButton;    btn_photo: TButton;    btn_save: TButton;    btn_cancel: TButton;
    cg_answers: TCheckGroup;    Label1: TLabel;    PhotoOpen: TOpenDialog;    question_edit: TEdit;
    GroupBox1: TGroupBox;
    procedure answer_countChange(Sender: TObject);
    procedure answer_editClick(Sender: TObject);
    procedure btn_cancelClick(Sender: TObject);
    procedure btn_explanationClick(Sender: TObject);
    procedure btn_photoClick(Sender: TObject);
    procedure btn_saveClick(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: boolean);
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure question_editChange(Sender: TObject);
  private
    Help : HelpRec;
```

```
  PhotoPath : String;
  NeedToSave : boolean;
 public
  Task : TaskRec;
 end;
var
 TaskManyOfManyWnd: TTaskManyOfManyWnd;
implementation
{$R *.lfm}
procedure TTaskManyOfManyWnd.btn_saveClick(Sender: TObject);
var
 i : integer;
 haschecked : boolean;
begin
 haschecked := false;
 for i := 0 to answer_count.ItemIndex do
  if (cg_answers.Checked[i]) then
  begin
   haschecked := true;
   break;
  end;
 if(haschecked) then
 begin
  Task.Question.Text     := question_edit.Text;
  Task.Question.PhotoPath := PhotoPath;
  Task.Help              := Help;
  Task.AnswersCount      := answer_count.ItemIndex + 1;
  for i := 1 to Task.AnswersCount do
   begin
    Task.Answers[i].IsRight := cg_answers.Checked[i - 1];
    Task.Answers[i].Text    := cg_answers.Items.Strings[i - 1];
   end;
  NeedToSave  := False;
  TaskManyOfManyWnd.Close;
 end
 else
  MessageDlg('Збереження даних',
   'Не можливо збрегти дані!' + #13#10 +'Не було обрано жодної відповіді, як правильної.' + #13#10 +
    'Повинна бути обрана хоча б одна відповідь як правильна.',     mtError,     [mbOk],     0);
```

```
end;
procedure TTaskManyOfManyWnd.FormCloseQuery(Sender: TObject;
 var CanClose: boolean);
var
 res : TModalResult;  i : integer;  save : boolean;
begin
 save := false;
 for i := 0 to answer_count.ItemIndex do
  if (cg_answers.Checked[i] <> Task.Answers[i + 1].IsRight) then
  begin
   save := true;
   break;
  end;
 if (NeedToSave) or (save) then
 begin
  res := MessageDlg('Збереження даних',     'Дані в завданні були змінені. Ви бажаєте зберегти?',
   mtConfirmation,     [mbYes, mbNo, mbCancel],     0);
  case res of
  mrYes:
  begin
   btn_saveClick(Self);
   CanClose := true;
  end;
  mrNo:    CanClose := true;
  mrCancel: CanClose := false;
  end;
 end;
end;
procedure TTaskManyOfManyWnd.FormCreate(Sender: TObject);
var
 index : integer;
begin
 for index := 0 to cg_answers.Items.Count - 1 do
  cg_answers.Controls[index].PopupMenu := AnswersPopupMenu;
end;
procedure TTaskManyOfManyWnd.FormShow(Sender: TObject);
var
 i : integer;
begin
```

```
 answer_count.ItemIndex  := Task.AnswersCount - 1;
 question_edit.Text      := Task.Question.Text;
 PhotoPath               := Task.Question.PhotoPath;
 Help                    := Task.Help;
 for i := 0 to cg_answers.Items.Count - 1 do
  if (i < Task.AnswersCount) then
  begin
   cg_answers.Items.Strings[i]     := Task.Answers[i + 1].Text;
   cg_answers.Controls[i].Enabled  := true;
   cg_answers.Checked[i]           := Task.Answers[i + 1].IsRight;
  end
  else
  begin
   cg_answers.Items.Strings[i] := 'Відповідь № ' + IntToStr(i + 1);
   cg_answers.Controls[i].Enabled := false;
  end;
 NeedToSave  := False;
end;
procedure TTaskManyOfManyWnd.question_editChange(Sender: TObject);
begin  NeedToSave  := True;end;
procedure TTaskManyOfManyWnd.btn_cancelClick(Sender: TObject);
begin
 NeedToSave  := False;
 TaskManyOfManyWnd.Close;
end;
procedure TTaskManyOfManyWnd.answer_countChange(Sender: TObject);
var
 i : integer;
begin
 if(answer_count.ItemIndex <> -1) then
 begin
  for i := 0 to answer_count.ItemIndex do
   cg_answers.Controls[i].Enabled := true;
  for i := answer_count.ItemIndex + 1 to cg_answers.Items.Count - 1 do
  begin
   cg_answers.Controls[i].Enabled := false;
   cg_answers.Checked[i] := false;
  end;
  NeedToSave  := True;
```

```
  end;
end;
procedure TTaskManyOfManyWnd.answer_editClick(Sender: TObject);
var
  value : string;
  index : integer;
begin
  for index := 0 to cg_answers.Items.Count - 1 do
    if (AnswersPopupMenu.PopupComponent = cg_answers.Controls[index]) then
      break;
  repeat
    value := InputBox('Відповідь №'+IntToStr(index + 1),            'Введіть нове значення для відповіді',
           cg_answers.Items.Strings[index]);
    if (value = '') then
      MessageDlg('Помилка вводу',     'Відповідь не може бути пустою!',     mtError,     [mbOk],     0);
  until value <> '';
  cg_answers.Items.Strings[index] := value;
  NeedToSave  := True;
end;
procedure TTaskManyOfManyWnd.btn_explanationClick(Sender: TObject);
begin
  ExplanationWnd.Help := Help;
  ExplanationWnd.ShowModal;
  Help       := ExplanationWnd.Help;
  NeedToSave  := True;
end;
procedure TTaskManyOfManyWnd.btn_photoClick(Sender: TObject);
begin
 if (PhotoPath <> 'null') then
  begin
    if( MessageDlg('Видалення фото',     'До завдання вже прив''язане фото. Ви бажаєте його видалити?',
      mtConfirmation,     [mbYes, mbNo],     0) = mrYes) then
       PhotoPath := 'null';
  end
  else if (PhotoOpen.Execute) then
    PhotoPath := PhotoOpen.FileName;
  NeedToSave := True;
end;
end.
```

Файл task_match_wnd.pas

```
unit task_match_wnd;
{$mode objfpc}{$H+}
interface
uses
 Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, Grids, Menus,
 explanation_wnd, exercise_unit;
type
 TTaskMatchWnd = class(TForm)
  AnswersPopupMenu: TPopupMenu;  answer_edit: TMenuItem;  btn_cancel: TButton;
  btn_explanation: TButton;  btn_photo: TButton;  btn_save: TButton;  GroupBox1: TGroupBox;
  PhotoOpen: TOpenDialog;  qa_grid: TStringGrid;
  procedure answer_editClick(Sender: TObject);
  procedure btn_cancelClick(Sender: TObject);
  procedure btn_explanationClick(Sender: TObject);
  procedure btn_photoClick(Sender: TObject);
  procedure btn_saveClick(Sender: TObject);
  procedure FormCloseQuery(Sender: TObject; var CanClose: boolean);
  procedure FormCreate(Sender: TObject);
  procedure FormShow(Sender: TObject);
  procedure qa_gridSetEditText(Sender: TObject; ACol, ARow: Integer;
   const Value: string);
 private
  Help : HelpRec;  PhotoPath : String;  NeedToSave : boolean;
 public
  Task : TaskRec;
 end;
var
 TaskMatchWnd: TTaskMatchWnd;
implementation
{$R *.lfm}
procedure TTaskMatchWnd.btn_photoClick(Sender: TObject);
begin
 if (PhotoPath <> 'null') then
  begin
   if( MessageDlg('Видалення фото',  'До завдання вже прив''язане фото. Ви бажаєте його видалити?',
    mtConfirmation,      [mbYes, mbNo],      0) = mrYes) then
     PhotoPath := 'null';
```

```
    end
    else if (PhotoOpen.Execute) then
      PhotoPath := PhotoOpen.FileName;
  NeedToSave := True;;
end;
procedure TTaskMatchWnd.btn_saveClick(Sender: TObject);
var
 i : integer;  haschecked : boolean;
begin
 haschecked := false;
 for i := 1 to qa_grid.RowCount - 1 do
  if (qa_grid.Cells[0, i] = 'true') then
  begin
   haschecked := true;
   break;
  end;
 if (haschecked) then
 begin
    haschecked := true;
    for i := 1 to qa_grid.RowCount - 1 do
     if (qa_grid.Cells[0, i] = 'true') and       ((qa_grid.Cells[1, i] = '') or          (qa_grid.Cells[2, i] = '')) then
     begin
       haschecked := true;
       break;
     end;
    if (haschecked) then
    begin
     Task.Question.Text     := 'Встановіть відповідність';
     Task.Question.PhotoPath := PhotoPath;
     Task.Help              := Help;
     for i := 1 to qa_grid.RowCount - 1 do
       if(qa_grid.Cells[1, i] <> '') and          (qa_grid.Cells[2, i] <> '') then
       begin
         if qa_grid.Cells[0, i] = 'Так' then
           Task.Answers[i].IsRight    := true
         else
           Task.Answers[i].IsRight    := false;
         Task.Answers[i].LabelAnswer := qa_grid.Cells[1, i];
         Task.Answers[i].Text       := qa_grid.Cells[2, i];
```

```
        end;
       NeedToSave  := False;
       TaskMatchWnd.Close;
     end
     else
       MessageDlg('Збереження даних',         'Не можливо збрегти дані!' + #13#10 +
        'Одна з правильних відповідей не повна.',         mtError,         [mbOk],         0);
  end
  else
    MessageDlg('Збереження даних',
     'Не можливо збрегти дані!' + #13#10 +'Не було обрано жодної відповіді, як правильної.' + #13#10 +
      'Повинна бути обрана хоча б одна відповідь як правильна.',      mtError,      [mbOk],      0);
end;
procedure TTaskMatchWnd.FormCloseQuery(Sender: TObject; var CanClose: boolean);
var
  res : TModalResult;
begin
  if(NeedToSave) then
  begin
    res := MessageDlg('Збереження даних',      'Дані в завданні були змінені. Ви бажаєте зберегти?',
      mtConfirmation,      [mbYes, mbNo, mbCancel],      0);
    case res of
    mrYes:
    begin
      btn_saveClick(Self);
      CanClose := true;
    end;
    mrNo:     CanClose := true;
    mrCancel: CanClose := false;
    end;
  end;
end;
procedure TTaskMatchWnd.FormCreate(Sender: TObject);
begin
  qa_grid.Cells[0, 0]   := 'Враховувати';
  qa_grid.Cells[1, 0]   := 'Питання';
  qa_grid.Cells[2, 0]   := 'Відповідь';
  qa_grid.ColWidths[0]  := 80;
  qa_grid.ColWidths[1]  := 160;
```

```
  qa_grid.ColWidths[2]  := 160;
end;
procedure TTaskMatchWnd.FormShow(Sender: TObject);
var
  i : integer;
begin
  for i := 1 to Task.AnswersCount do
  begin
    if (Task.Answers[i].IsRight) then
      qa_grid.Cells[0, i] := 'Так'
    else
      qa_grid.Cells[0, i] := 'Hi';
    qa_grid.Cells[1, i] := Task.Answers[i].LabelAnswer;
    qa_grid.Cells[2, i] := Task.Answers[i].Text;
  end;
  PhotoPath   := Task.Question.PhotoPath;
  Help        := Task.Help;
  NeedToSave  := False;
end;
procedure TTaskMatchWnd.qa_gridSetEditText(Sender: TObject; ACol,
  ARow: Integer; const Value: string);
begin  NeedToSave  := True;end;
procedure TTaskMatchWnd.btn_explanationClick(Sender: TObject);
begin
  ExplanationWnd.Help := Help;
  ExplanationWnd.ShowModal;
  Help        := ExplanationWnd.Help;
  NeedToSave  := True;
end;
procedure TTaskMatchWnd.btn_cancelClick(Sender: TObject);
begin  NeedToSave := False;  TaskMatchWnd.Close;end;
procedure TTaskMatchWnd.answer_editClick(Sender: TObject);
begin
  if (qa_grid.Row <> -1) and (qa_grid.Row <> 0) then
  begin
    if (qa_grid.Cells[0, qa_grid.Row] = 'Так') then
      qa_grid.Cells[0, qa_grid.Row] := 'Hi'
    else
      qa_grid.Cells[0, qa_grid.Row] := 'Так';
```

```
    NeedToSave  := True;
  end;
end;
end.
```

Файл task_oneofmany_wnd.pas

```
unit task_oneofmany_wnd;
{$mode objfpc}{$H+}
interface
uses
  Classes, SysUtils, Forms, Controls, Graphics, Dialogs, ExtCtrls, StdCtrls,
  Menus, explanation_wnd, exercise_unit;
type
  TTaskOneOfManyWnd = class(TForm)
    answer_count: TComboBox;    btn_save: TButton;    btn_cancel: TButton;    btn_photo: TButton;
    btn_explanation: TButton;    Label1: TLabel;    answer_edit: TMenuItem;    PhotoOpen: TOpenDialog;
    AnswersPopupMenu: TPopupMenu;    question_edit: TEdit;    GroupBox1: TGroupBox;
    rg_answers: TRadioGroup;
    procedure answer_countChange(Sender: TObject);
    procedure answer_editClick(Sender: TObject);
    procedure btn_cancelClick(Sender: TObject);
    procedure btn_explanationClick(Sender: TObject);
    procedure btn_photoClick(Sender: TObject);
    procedure btn_saveClick(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: boolean);
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure question_editChange(Sender: TObject);
  private
    Help : HelpRec;
    PhotoPath : String;
    NeedToSave : boolean;
  public
    Task : TaskRec;
  end;
var
  TaskOneOfManyWnd: TTaskOneOfManyWnd;
implementation
{$R *.lfm}
```

```
procedure TTaskOneOfManyWnd.btn_saveClick(Sender: TObject);
var
 i : integer;
begin
 Task.Question.Text     := question_edit.Text;
 Task.Question.PhotoPath := PhotoPath;
 Task.Help              := Help;
 Task.AnswersCount      := answer_count.ItemIndex + 1;
 for i := 1 to Task.AnswersCount do
  begin
   Task.Answers[i].IsRight := false;
   Task.Answers[i].Text    := rg_answers.Items.Strings[i - 1];
  end;
 Task.Answers[rg_answers.ItemIndex + 1].IsRight := true;
 NeedToSave  := False;
 TaskOneOfManyWnd.Close;
end;
procedure TTaskOneOfManyWnd.FormCloseQuery(Sender: TObject;
 var CanClose: boolean);
var
 res : TModalResult;
begin
 if(NeedToSave) or
  not(Task.Answers[rg_answers.ItemIndex + 1].IsRight)
  then
 begin
  res := MessageDlg('Збереження даних',
   'Дані в завданні були змінені. Ви бажаєте зберегти?',
   mtConfirmation,
   [mbYes, mbNo, mbCancel],
   0);
  case res of
  mrYes:
  begin
   btn_saveClick(Self);
   CanClose := true;
  end;
  mrNo:    CanClose := true;
  mrCancel: CanClose := false;
```

```
   end;
  end;
end;
procedure TTaskOneOfManyWnd.FormCreate(Sender: TObject);
var
  index : integer;
begin
  for index := 0 to rg_answers.Items.Count - 1 do
    rg_answers.Controls[index].PopupMenu := AnswersPopupMenu;
end;
procedure TTaskOneOfManyWnd.FormShow(Sender: TObject);
var
  i : integer;
begin
  answer_count.ItemIndex  := Task.AnswersCount - 1;
  question_edit.Text      := Task.Question.Text;
  PhotoPath               := Task.Question.PhotoPath;
  Help                    := Task.Help;
  for i := 0 to rg_answers.Items.Count - 1 do
  begin
    if (i < Task.AnswersCount) then
    begin
      rg_answers.Items.Strings[i] := Task.Answers[i + 1].Text;
      rg_answers.Controls[i].Enabled  := true;
      if (Task.Answers[i + 1].IsRight) then
        rg_answers.ItemIndex := i;
    end
    else
    begin
      rg_answers.Items.Strings[i] := 'Відповідь № ' + IntToStr(i + 1);
      rg_answers.Controls[i].Enabled := false;
    end;
  end;
  NeedToSave  := False;
end;
procedure TTaskOneOfManyWnd.question_editChange(Sender: TObject);
begin  NeedToSave  := True;end;
procedure TTaskOneOfManyWnd.btn_cancelClick(Sender: TObject);
begin
```

```
  NeedToSave  := False;
  TaskOneOfManyWnd.Close;
end;
procedure TTaskOneOfManyWnd.answer_countChange(Sender: TObject);
var
  i : integer;
begin
  if(answer_count.ItemIndex <> -1) then
  begin
    for i := 0 to answer_count.ItemIndex do
      rg_answers.Controls[i].Enabled := true;
    for i := answer_count.ItemIndex + 1 to rg_answers.Items.Count - 1 do
      rg_answers.Controls[i].Enabled := false;
    rg_answers.ItemIndex := 0;
    NeedToSave  := True;
  end;
end;
procedure TTaskOneOfManyWnd.answer_editClick(Sender: TObject);
var
  value : string;
  index : integer;
begin
  for index := 0 to rg_answers.Items.Count - 1 do
    if (AnswersPopupMenu.PopupComponent = rg_answers.Controls[index]) then
      break;
  repeat
    value := InputBox('Відповідь №'+IntToStr(index + 1),          'Введіть нове значення для відповіді',
           rg_answers.Items.Strings[index]);
    if (value = '') then
      MessageDlg('Помилка вводу',    'Відповідь не може бути пустою!',    mtError,    [mbOk],    0);
  until value <> '';
  rg_answers.Items.Strings[index] := value;
  NeedToSave  := True;
end;
procedure TTaskOneOfManyWnd.btn_explanationClick(Sender: TObject);
begin
  ExplanationWnd.Help := Help;
  ExplanationWnd.ShowModal;
  Help       := ExplanationWnd.Help;
```

```
 NeedToSave  := True;
end;
procedure TTaskOneOfManyWnd.btn_photoClick(Sender: TObject);
begin
 if (PhotoPath <> 'null') then
 begin
  if( MessageDlg('Видалення фото',     'До завдання вже прив''язане фото. Ви бажаєте його видалити?',
    mtConfirmation,      [mbYes, mbNo],      0) = mrYes) then
      PhotoPath := 'null';
 end
 else if (PhotoOpen.Execute) then
  PhotoPath := PhotoOpen.FileName;
 NeedToSave := True;
end;
end.
```

Файл explanation_wnd.pas

```
unit explanation_wnd;
{$mode objfpc}{$H+}
interface
uses
 Classes, SysUtils, Forms, Controls, Graphics, Dialogs, StdCtrls, exercise_unit;
type
 TExplanationWnd = class(TForm)
  btn_save: TButton;    btn_cancel: TButton;    btn_photo: TButton;    hint_edit: TEdit;
  GroupBox1: TGroupBox;   solution_memo: TMemo;   GroupBox2: TGroupBox;  PhotoOpen: TOpenDialog;
  procedure btn_cancelClick(Sender: TObject);
  procedure btn_photoClick(Sender: TObject);
  procedure btn_saveClick(Sender: TObject);
  procedure FormCloseQuery(Sender: TObject; var CanClose: boolean);
  procedure FormShow(Sender: TObject);
  procedure hint_editChange(Sender: TObject);
  procedure solution_memoChange(Sender: TObject);
 private
  PhotoPath : String;
  NeedToSave : boolean;
 public
  Help : HelpRec;
 end;
```

```
var
  ExplanationWnd: TExplanationWnd;
implementation
{$R *.lfm}
procedure TExplanationWnd.btn_saveClick(Sender: TObject);
begin
  Help.Hint      := hint_edit.Text;
  Help.Solution   := solution_memo.Text;
  Help.PhotoPath  := PhotoPath;
  NeedToSave     := False;
  ExplanationWnd.Close;
end;
procedure TExplanationWnd.FormCloseQuery(Sender: TObject; var CanClose: boolean
  );
var
  res : TModalResult;
begin
  if(NeedToSave) then
  begin
    res := MessageDlg('Збереження даних',     'Дані були змінені. Ви бажаєте зберегти?',
      mtConfirmation,     [mbYes, mbNo, mbCancel],     0);
    case res of
    mrYes:
    begin
      btn_saveClick(Self);
      CanClose := true;
    end;
    mrNo:     CanClose := true;
    mrCancel: CanClose := false;
    end;
  end;
end;
procedure TExplanationWnd.FormShow(Sender: TObject);
var
  s : string;
  p : integer;
begin
  solution_memo.Clear;
  s := Help.Solution;
```

```
  p := Pos(#10, s);
  while p <> 0 do
  begin
    if (s <> '') then
      solution_memo.Lines.Add(Copy(s, 1, p));
    Delete(s, 1, p);
    p := Pos(#10, s);
  end;
  if (s <> '') then
    solution_memo.Lines.Add(s);
  hint_edit.Text  := Help.Hint;
  PhotoPath       := Help.PhotoPath;
  NeedToSave      := False;
end;
procedure TExplanationWnd.btn_photoClick(Sender: TObject);
begin
  if (PhotoPath <> 'null') then
  begin
    if( MessageDlg('Видалення фото',     'До завдання вже прив''язане фото. Ви бажаєте його видалити?',
      mtConfirmation,      [mbYes, mbNo],      0) = mrYes) then
        PhotoPath := 'null';
  end
  else if (PhotoOpen.Execute) then
    PhotoPath := PhotoOpen.FileName;
  NeedToSave := True;
end;
procedure TExplanationWnd.btn_cancelClick(Sender: TObject);
begin
  NeedToSave := False;
  ExplanationWnd.Close;
end;
procedure TExplanationWnd.hint_editChange(Sender: TObject);
begin  NeedToSave := True;end;
procedure TExplanationWnd.solution_memoChange(Sender: TObject);
begin  NeedToSave := True;end;
end.
```