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

«_____»_____2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему

СТВОРЕННЯ АЛГОРИТМУ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТРЕНАЖЕРА ПОБУДОВИ ВИПЕРЕДЖЕНОЇ НОРМАЛЬНОЇ ФОРМИ ДИСТАНЦІЙНОГО КУРСУ «МАТЕМАТИЧНА ЛОГІКА»

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Коровайкін Єгор Дмитрович
_____«___»_____2021р.

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

ПОЛТАВА 2021 р.

РЕФЕРАТ

Записка: 52 стор., основна частина 40 стор., джерел – 10.

Предмет розробки – тренажер для навчання темі «Випереджена нормальна форма в логіці предикатів»

Мета роботи – програмно реалізувати тренажер для навчання темі «Випереджена нормальна форма в логіці предикатів»

Методи,які були використані для розв'язування задачі – для створення програмного забезпечення була використана платформа Microsoft Visual Studio та мова програмування C#.

Розроблено алгоритм тренажера та здійснена програмна реалізація тренажера «Випереджена нормальна форма в логіці предикатів»

Ключові слова: ВИПЕРЕДЖЕНА НОРМАЛЬНА ФОРМА , ЛОГІКА ПРЕДИКАТІВ, НАВЧАЛЬНИЙ ТРЕНАЖЕР, ДИСТАНЦІЙНЕ НАВЧАННЯ.

ЗМІСТ

Вступ

Метою проекту є розробка тренажеру з теми «Випереджена нормальна форма в логіці предикатів» для дистанційного курсу «Математична логіка» , завдяки якому студенти зможуть перевірити свої знання та побачити недоліки у вивченні теми. Для досягнення мети в ході роботи необхідно виконати такі завдання

- розглянути матеріал за темою роботи;
- розроблення алгоритму тренажеру;
- вибір мови програмування;
- програмна реалізація тренажеру;
- перевірка роботи тренажеру
- описати роботу тренажеру

Об'єкт розробки - це процес розроблення тренажеру для дистанційного курсу «Математична логіка».

Предмет розробки - тренажер для дистанційного курсу «Математична логіка» та програма розроблена на його основі. Методи розробки. Випереджена нормальна форма для розробки алгоритму. Microsoft Visual Studio та об’єктно-орієнтована мова програмування C# для реалізації тренажеру.

1. Постановка задачі

Для виконання роботи треба розробити тренажер з теми «Випереджена нормальна форма в логіці предикатів».

Програма-тренажер повинна мати:

- Стартову сторінку
- Сторінки з питаннями
- Повідомлення про помилки
- Сторінку з результатами проходження завдань
- Показ правильної відповіді після того як відповів користувач

Проектування тренажеру складається з наступних кроків:

Крок 1. Огляд недоліків та переваг попередніх робіт

Крок 2. Формування вимог до тренажера.

Крок 3. Вибір середовища та мови програмування.

Крок 4. Розробити блок-схеми роботи тренажера

Крок 5. Програмна реалізація на обраній мові програмування з урахуванням всіх вимог до тренажера.

Крок 6. Тестування та вдосконалення розробленого програмного продукту. На цьому кроці потрібно повністю перевірити програму, її складові частини, роботу в середовищах. За потребою можливо вдосконалити тренажер, додавши нові можливості чи компоненти.

Для розробки тренажеру була обрана мова програмування C# у середовищі розробки Microsoft Visual Studio .

2.Інформаційний огляд

2.1 Огляд робіт

В процесі написання бакалаврської роботи було розглянуто наступні тренажери: Алієва Ф. П. «Матриці і визначники» який був написаний на мові програмування Delphi та Прімова Х.Н. «Метод Крамера» який був написаний також на мові програмування Delphi. Після запуску тренажерів на екрані з’являється зручний інтерфейс з вибором тем для проходження.

2.2 Позитивні аспекти оглянутих робіт

Позитивні аспекти оглянутих робіт

1. Зручний інтерфейс
2. Завдання описані достатньо зручним шрифтом
3. Немало теоретичних та практичних питань, також існують підказки, якщо користувач вибрав неправильну відповідь.

2.3 Вади розробок оглянутих робіт

Вади розробок оглянутих робіт

1. Після того, як користувач вибрав неправильну відповідь, з’являється повідомлення з правильною відповіддю та автоматично відбуваються перехід до наступного питання без можливості повторно відповісти.
2. Якщо користувач обрав неправильну відповідь в питаннях, де потрібно вписати значення у комірку, відбувається автоматичне заповнення поточної клітини, тому є можливість проходження тренажеру натискаючи лише одну кнопку.

3.Теоретична частина

3.1 Алгоритм роботи тренажера по зведенню до випередженої нормальної форми

На екрані:

1 крок. Виберіть одну правильну відповідь. Формула логіки предикатів задана у випередженій нормальній формі , якщо вона має вигляд:

- $Q_1x_1\ v\ Q_2x_2\ v\ \ldots\ v\ Q_nx_n\ M$, де кожне $Q_1x_1\ Q_2x_2\ \ldots Q_nX_n$(i=1, 2, … ,n) – це $\forall x_i$ або $\exists x_i$, а формула М не містить предикатів

- Q_1x_1 v Q_2x_2v ... v $Q_nx_n\ M$, де кожне $Q_1x_1\ Q_2x_2\ \ldots Q_nX_n$(i=1, 2, … ,n) – це $\forall x_i$ або $\exists x_i$, а М – це формула, що не містить предикатів

- $Q_1x_1\ Q_2x_2\ \ldots\ Q_nx_n\ M$, де кожне $Q_1x_1\ Q_2x_2\ \ldots Q_nX_n\ (i = \overline{1,n}\ )$, а М – це формула, яка може містити квантори

- $Q_1x_1\ Q_2x_2\ \ldots\ Q_nx_n \rightarrow\ M$, де кожне $Q_1x_1\ Q_2x_2\ \ldots Q_nX_n\ (i = \overline{1,n}\ )$, – це $\exists x_i$, а М – це формула, що містить квантори $\forall$

Текстове повідомлення про помилку: «Формула логіки предикатів записана не правильно»

2 крок. У формулі $F = (Q_1x_nQ_2x_2 \cdots Q_nx_nM)$,

$Q_1x_nQ_2x_2 \cdots Q_nx_n$ називається [▽] ,

а М -[▽] формули F.

[▽] - випадаючий список з трьома назвами: «суфіксом» , «префіксом» та «матрицею» , вибрати користувачу.

Текст повідомлення про помилку: «У формулі $F = (Q_1x_n \ldots Q_nx_n)$, $Q_1x_nQ_2x_2 \cdots Q_nx_n$ називається префіксом , а М – матрицею формули F

Виберіть правильні відповіді

Формули , що задані у ВНФ:

-$\forall_x \exists_y (P(x) \text{v} Q(y)$

- $\forall_y (P(x) \rightarrow \exists_y Q(y)$

- $\exists_x \forall_y \exists_z \left(\left(A(x,y) \text{v} Q(y,z) \right) \wedge \overline{R(z))} \right.$

-$\forall_x P(x) \leftrightarrow (\overline{\exists_x \forall_y Q(x,y))}$

Текст повідомлення про помилку «правильна відповідь $\forall_x \exists_y (P(x) \text{v} Q(y)$ та $\exists_x \forall_y \exists_z ((A(x,y) \text{v} Q(y,z)) \Lambda \overline{R(z))}$ »

4 крок. Введіть значення у комірки які відповідають номерам кроків алгоритму зведення формулі до ВНФ

- Винести квантори на початок формули , використовуючи відповідні закони
- Опустити знаки операцій безпосередньо на предикати , використовуючи закон подвійного заперечення і закони де Моргана
- Перейменування зв’язаних змінних при необхідності
- Виключити логічні зв’язки еквівалентності (∼) та імплікації (→) , виразивши їх через операції диз’юнкції , кон’юнкції та заперечення

Текст повідомлення про помилку «алгоритм введення до ВНФ :

1. Виключити логічні зв’язки еквівалентності (∼) та імплікації (→) , виразивши їх через операції диз’юнкції , кон’юнкції та заперечення
2. Опустити знаки операцій безпосередньо на предикати , використовуючи закон подвійного заперечення і закони де Моргана
3. Перейменування зв’язаних змінних при необхідності

4. Винести квантори на початок формули , використовуючи відповідні закони

Крок 5. Виберіть правильні варіанти для усунення у формулі еквівалентності та імплікації (↔ та →)?

- P ↔ Q = (P→Q) ∧ (Q→P)

- P→Q = $\bar{P}$ v Q

- P ↔ Q =(P ∧ Q) → (P v Q)

- P→Q = P v $\bar{Q}$

Текст повідомлення про помилку « Правильні відповіді: P ↔ Q = (P→Q) ∧ (Q→P) та P→Q = $\bar{P}$ v Q »

Крок 6. Виберіть правильні варіанти.

Опустити знаки операції заперечення безпосередньо на предикати і використовуються наступні закони:

- $\overline{F \text{v} G} = F \wedge G$
- $\overline{F \text{v} G} = \bar{F} \wedge \bar{G}$
- $\overline{F \text{v} G} = (F \; v \; \bar{G})(\bar{F} \; v \; G)$
- $\bar{\bar{F}} = F$
- F v G = $\bar{F} \vee G$
- $\overline{F \wedge G} = \bar{F} \vee \bar{G}$
- $\overline{\exists_x} F(x) = \forall_x \overline{F(x)}$
- $\overline{\forall_x} F(x) = \exists_x \overline{F(x)}$
- $\overline{\exists_x} F(x) = \exists x \overline{F(x)}$

Текст повідомлення про помилку « Правильні відповіді:

- $\overline{F \vee G} = \bar{F} \wedge \bar{G}$
- $\bar{\bar{F}} = F$
- $\overline{F \wedge G} = \bar{F} \vee \bar{G}$
- $\overline{\exists_x} F(x) = \forall_x \overline{F(x)}$
- $\overline{\exists_x} F(x) = \exists_x \overline{F(x)}$

Крок 7. Після усунення імплікації формула $\forall_x P(x) \rightarrow \exists_x Q(y)$ набуде вигляду :

- $\overline{\forall_x P(x)} \; v \; \exists_y Q(y)$
- $\forall_x P(x) \; v \; \overline{\exists_y Q(y)}$
- $\forall_x P(x) \wedge \overline{\exists_y Q(y)}$
- $\overline{\forall_x P(x)} \wedge \exists_y Q(y)$

Повідомлення – підказка: «Використайте формулу $a \rightarrow b = \bar{a} \; v \; b$, вважаючи що $a = \forall_x P(x)$, а $b = \exists_x Q(y)$ »

Текст повідомлення про помилку « Правильна відповідь $\overline{\forall_x P(x)} \; v \; \exists_y Q(y)$ »

8 крок. Опускаємо знак операції заперечення у формулі:

$$\overline{\forall P(x)} \text{ v } \exists y \; Q(y)$$

Приймає вигляд

- $\forall_x P(x)$ v $\exists_y Q(y)$
- $\exists_x \overline{P(x)}$ v $\exists_y Q(y)$
- $\exists_x P(x)$ v $\exists_y Q(y)$
- $\forall_x \overline{P(x)}$ v $\exists_y Q(y)$

Текст повідомлення про помилку « Правильна відповідь - $\exists_x \overline{P(x)}$ v $\exists_y Q(y)$ »

9 крок. Яку з наступних формул можна використати до формули $\exists_x \overline{P(x)} \; v \; \exists_y Q(y)$ для зведення її до ВНФ?

- $\exists_x (C \; v \; P(x)) \; = \; C \; v \; \exists_x P(x)$

- $\exists_x (A(x) \; v \; B(x)) \; = \; \exists_x A(x) \; v \; \exists_x B(x)$

- $\exists_x A(x) \; v \; B(x) \; = \; \exists_x A(x) \; v \; \exists_x B(x)$

Повідомлення про помилку: « Правильна відповідь $\exists_x (C \; v \; P(x)) \; = \; C \; v \; \exists_x P(x)$ »

10 крок. Згідно $\exists_x (C \; v \; P(x) \; = \; C \; v \; \exists_x P(x)$, враховуючи , що $\exists_y Q(y) \; = \; C$, формула $\exists_x \overline{P(x)} \; v \; \exists_y Q(y)$ прийме вигляд:

	▽		▽		▽		▽		▽

У випадаючих списках: $\exists_x$;

$\forall_x$;

$P(x)$

$\overline{P(x)}$

$\overline{Q(x)}$

$Q(x)$

V

∧

$\exists_y$

Правильна відповідь: $\boxed{\exists_x}$ $\boxed{\overline{P(x)}}$ $\boxed{V}$ $\boxed{\exists_y}$ $\boxed{Q(y)}$

Після правильного введення на екрані з'являється : $\exists_x(\overline{P(x)}\ v\ \exists_y Q(y))$

11 крок.

Згідно $\exists_x(C\ v\ P(x)\ =\ C\ v\ \exists_x P(x)$ враховуючи , що $\overline{P(x)}\ =\ C$, формула $\exists_x(\overline{P(x)}\ v\ \exists_y Q(y))$ прийме вигляд:

У випадаючих списках: $\exists_x$

$\forall_x$

$P(x)$

$\overline{P(x)}$

$\overline{Q(x)}$

$Q(x)$

V

$\wedge$

$\exists_y$

Правильна відповідь: : $\boxed{\exists_x}$ $\boxed{\exists_y}$ $\boxed{\overline{P(x)}}$ $\boxed{\text{V}}$ $\boxed{Q(y)}$

Після правильного введення на екрані : $\exists_x\exists_y(\overline{P(x)}\ v\ Q(y))$ ВНФ для формули : $\forall_x P(x)\ \rightarrow\ \exists_y\ Q(y)$

Вивід кількості правильних відповідей.

Алгоритм завершено.

3.2. Обґрунтування вибору C#

C# - сучасна об’єктно-орієнована і типобезпечна мова програмування. C# дозволяє розробникам створювати безліч типів безпечних та надійних додатків , працюючих в системах .NET. C# відноситься до широко відомому сімейству мов C, та основних компонентів мови C# та більш ранніх версій.

C# - це об’єктно і компонентна-орієнтована мова програмування. C# надає мовні конструкції для безпосередньої підтримки такої концепції роботи. Завдяки цьому C# підходить для розробки і підтримки програмних компонентів. З моменту створення мови C# здобув безліч функцій для підтримки нових робочих навантажень та сучасними рекомендаціями по розробці програмного забезпечення.

Приклад декількох функцій мови C#, які дозволяють програмувати надійні та стійкі додатки:

Garbage collection (Прибирання сміття) – Автоматично звільняє пам’ять, зайняту недоступними невикористовуваними об’єктами.

Nullable types (Типи, допускаючи значення null) – забезпечує захист від змінних, які не посилаються на виділені об’єкти.

Exception handing (Обробка виключень) – надає структурований і розширюваний підхід до виявлення помилок і відновлення після них.

Lambda expression (лямда вирази) – підтримують прийоми функціонального програмування

Language Integrated Query (LINQ синтаксис) – Створює загальний шаблон для роботи з даними будь-якого джерела.

Підтримка мов для асинхронних операцій надає синтаксис для створення розподілених систем.

В C# діє «Єдина система типів». Усі типи включаючи типи-примітиви , такі як int та Bool, успадковують від одного кореневого типу Object. Усі типи використовують загальний набір операцій, а значення будь-якого типу можна зберігати , передавати і обробляти схожим чином. Більш того C# підтримує як визначаються користувачами посилальні типи, так і типи значень. C# дозволяє динамічно виділяти об’єкти та зберігати спрощені структури в стеці. C# підтримує універсальні методи і типи, що забезпечують підвищену безпеку типів і продуктивність# надає ітератори, які дозволяють розробникам класів колекцій визначати призначені для користувача варіанти поведінки для клієнтського коду.

В C# особлива увага приділяється контролю версій для забезпечення сумісності програм і бібліотек при їх зміні. Питання управління версіями істотно вплинули на такі аспекти розробки C#, як роздільні модифікатори.

Архітектура .NET

Програми C# виконуються в .NET , віртуальній системи виконання, викликаючи загальномовну середу виконання (CLR) та набір бібліотек класів. Середа CLR – це реалізація загальномовної інфраструктури мови (CLI), являючи міжнародним стандартом, від корпорації Майкрософт. CLI є основою для створення середовищ виконання і розробки, в яких мови і бібліотеки прозоро працюють один з одним.

Вихідний код, написаний на мові C # компілюється в проміжний мова (IL), який відповідає специфікаціям CLI. Код на мові IL і ресурси, в тому числі растрові зображення і рядки, зберігаються в збірці, зазвичай з розширенням .dll. Збірка містить маніфест з інформацією про типи, версіях, регіональних і регіональних параметрах для цієї збірки.

При виконанні програми C # збірка завантажується в середу CLR. Середа CLR виконує JIT-компіляцію з коду на мові IL в інструкції машинного мови. Середа CLR виконує інші операції, наприклад, автоматичну збірку

сміття, обробку винятків і управління ресурсами. Код, що виконується середовищем CLR, іноді називають «керованим кодом», щоб підкреслити відмінності цього підходу, відразу від «некерованого коду», який компілюється в машинний мову для певної платформи.

Забезпечення взаємодії між мовами є ключовою особливістю .NET. Код IL, створений компілятором C #, відповідає специфікації загальних типів (CTS). Код IL, створений з коду на C #, може взаємодіяти з кодом, створеним з версій .NET для мов F #, Visual Basic, C ++ і будь-яких інших з більш ніж 20 мов, сумісних з CTS. Одна збірка може містити кілька модулів, написаних на різних мовах .NET, і всі типи можуть посилатися один на одного, як якщо б вони були написані на одній мові.

На додаток до службам часу виконання .NET також включає розширені бібліотеки. Ці бібліотеки підтримують безліч різних робочих навантажень. Вони впорядковані по просторах імен, які надають різні корисні можливості: від операцій файлового введення і виведення до управління рядками і синтаксичного аналізу XML, від платформ веб-додатків до елементів управління Windows Forms. Зазвичай додаток C # активно використовують бібліотеку класів .NET для вирішення типових задач.

Отже, мова програмування C#, у якій поєднуються потужність і гнучкість універсальних мов програмування з високою ефективністю виконавчого коду й можливістю безпосереднього доступу до апаратних ресурсів комп’ютера – одна з найкращих мов програмування.

Для того щоб у усіх питань був однаковий інтерфейс був використаний один з методів ООП

Об`єктно-орієнтоване програмування— одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою.

Основу ООП складають чотири основні концепції: інкапсуляція, успадкування, поліморфізм та абстракція. Однією з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами).

Наслідування є одним з фундаментальних атрибутів об'єктно-орієнтованого програмування. Воно дозволяє визначити дочірній клас, який використовує (успадковує), розширює або змінює можливості батьківського класу. Клас, члени якого успадковуються, називається базовим класом. Клас, який успадковує члени базового класу, називається похідним класом.

C # і .NET підтримують тільки одиночне спадкоємство. Це означає, що кожен клас може успадковувати члени тільки одного класу. Але зате підтримується Транзитивне наслідування, яке дозволяє визначити ієрархію наслідування для набору типів. Іншими словами, тип D може успадковувати можливості типу C, який в свою чергу успадковує від типу B, який успадковує від базового класу A. Завдяки транзитивності наслідування члени типу A будуть доступні для типу D.

Клас або структура можуть реалізовувати один або кілька інтерфейсів. Реалізація інтерфейсів часто розглядається як метод для обходу обмежень одиночного наслідування або для реалізації наслідування структур. Але його основним призначенням є вираз зв'язку іншого роду між інтерфейсом і реалізують його типом. Цей зв'язок називається "can do" (може виконувати) і вона відрізняється від зв'язку наслідування. Інтерфейс визначає підмножина функцій (наприклад, перевірка рівності, порівняння і сортування об'єктів, або підтримка синтаксичного аналізу та форматування з урахуванням мови та регіональних параметрів). Інтерфейс надає ці функції всім типам, які його реалізують.

4.ПРАКТИЧНА ЧАСТИНА

4.1 Блок схема програми

На рис 4.1-4.4 зображені блок схемы алгоритму програми та її частин:

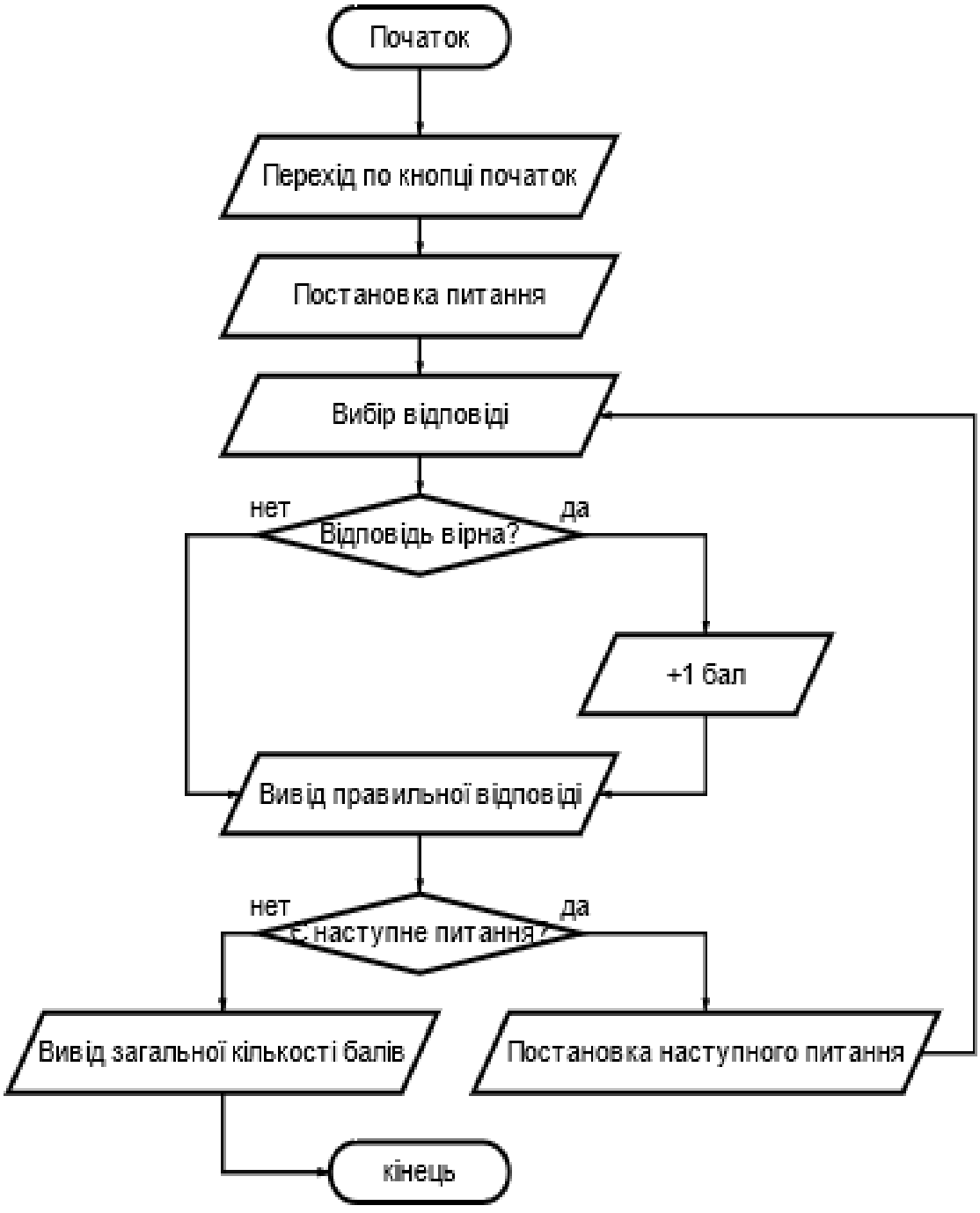

Рисунок 4.1 – Загальна блок-схема роботи програми-тренажера

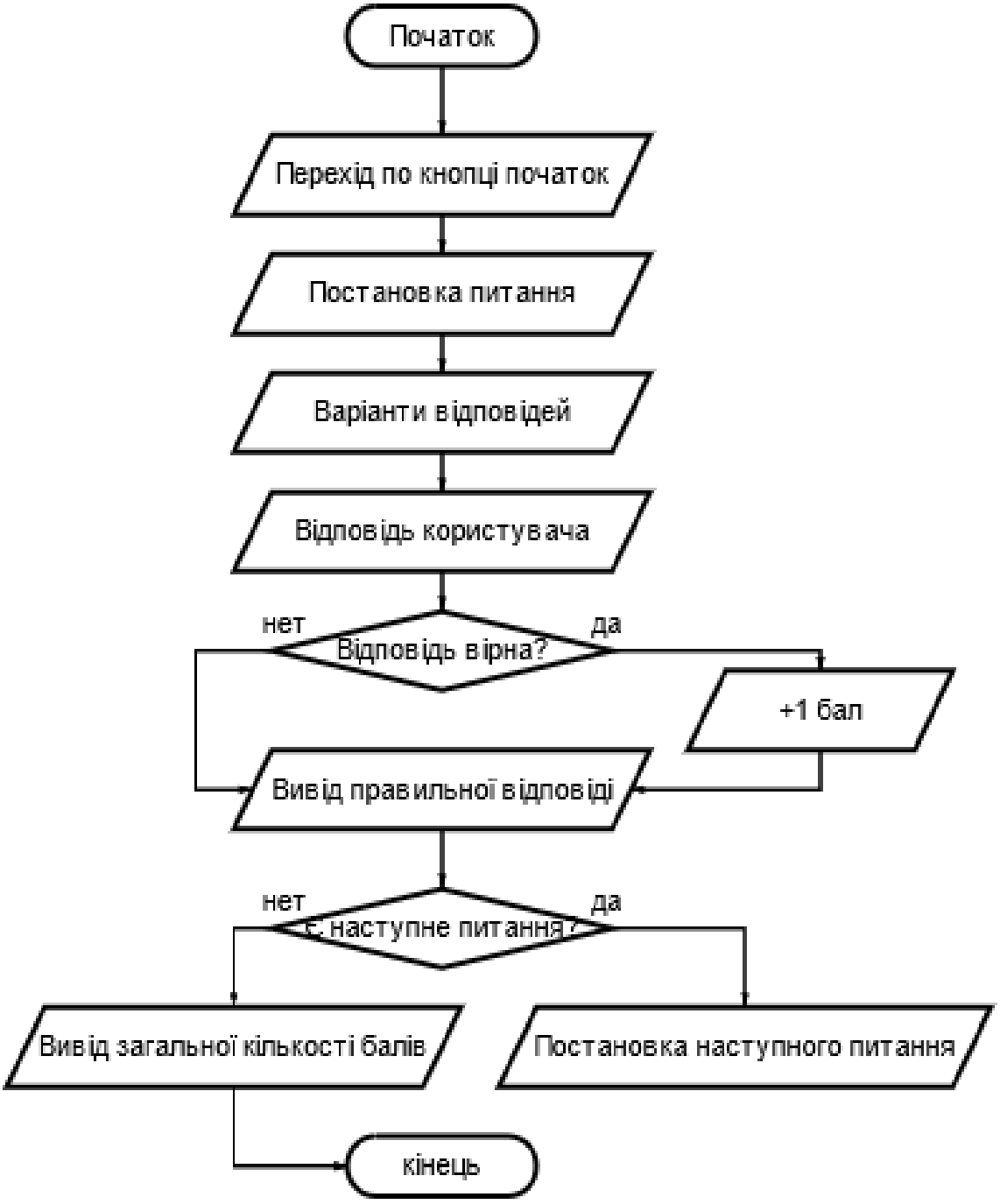

Рисунок 4.2 –Блок-схема питання з однією правильною відповідю

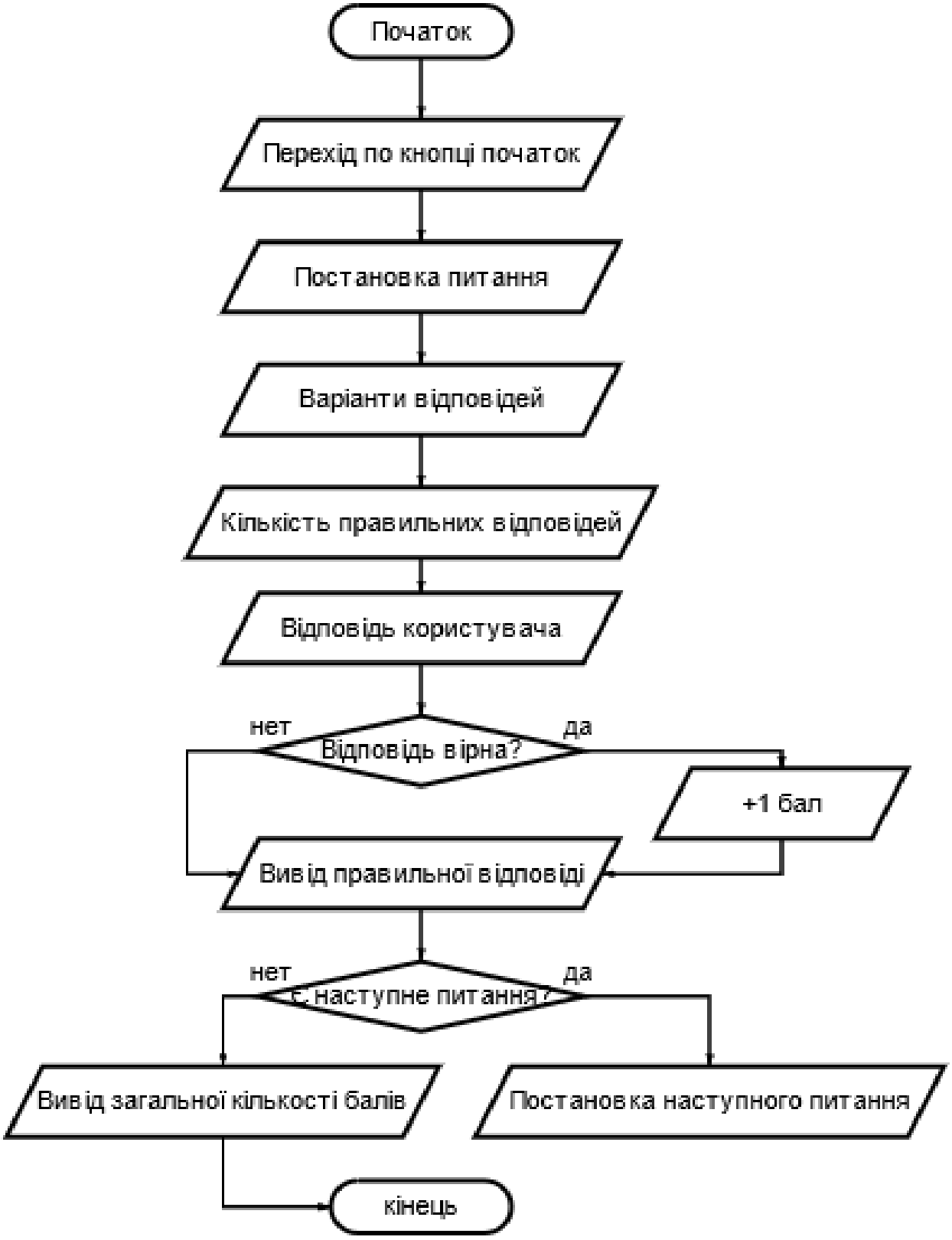

Рисунок 4.3 – Блок-схема питань з декількома правильними відповідями

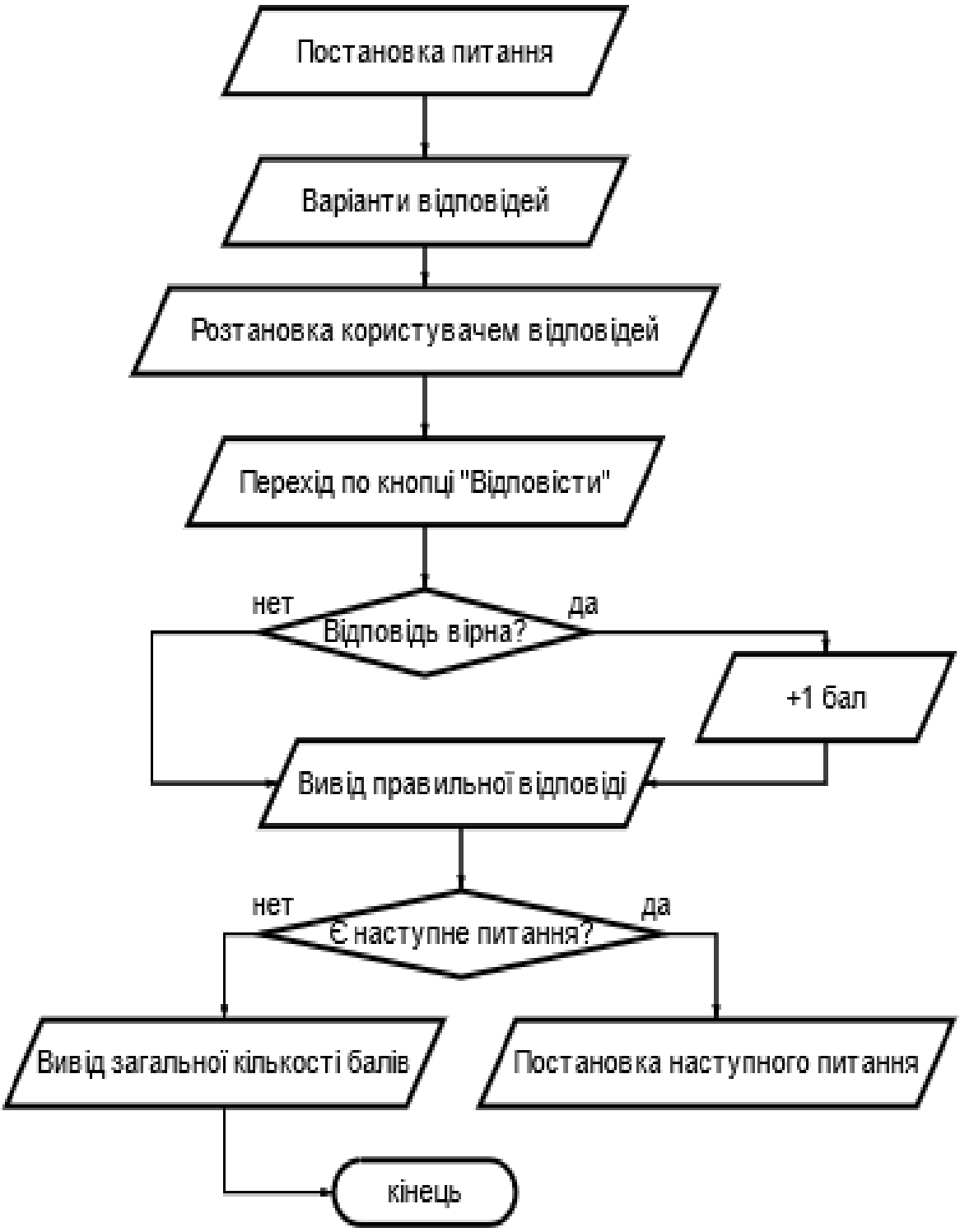

Рисунок 4.4 – Блок-схема питань з відповідями у випадаючих списках

4.2 Опис процесу програмної реалізації

Для розробки тренажеру з теми «Випереджена нормальна форма в логіці предикатів» було використано мову програмування C#, в середовищі Microsoft Visual Studio (рис 4.5).

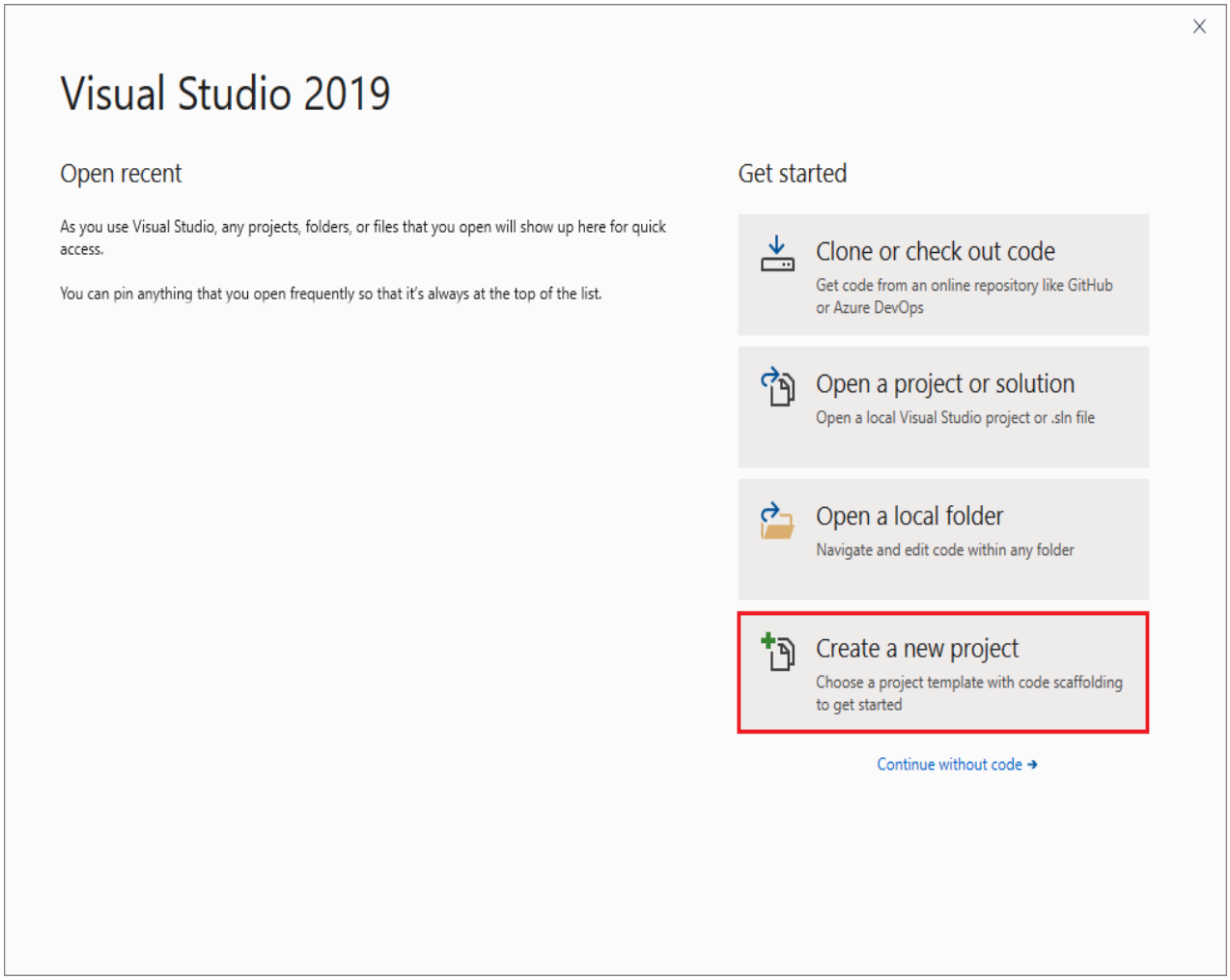

Рисунок 4.5 – Створення шаблону

При створенні проекту було обрано Windows Forms.Далі на форму додавалися елементи:

- Button – для створення кнопок;
- pictureBox – для вставлення рисунків у питаннях та відповідях
- TextBox, Label – для оформлення стартової сторінки;

На наступних рисунках буде наведено приклад програмної реалізації питань (рис 4.5)

```
public interface IQuiz
 {
     string Question { get; set; }
     Answer[] Answers { get; set; }
     decimal GetScore(params Answer[] choosenAnswer);
 }

 public class SingleAnswerQuiz : IQuiz
 {
     public string Question { get; set; }
     public string QuestionImage { get; set; }
     public Answer[] Answers { get; set; }
     private Answer[] CorrectAnswers => Answers.Where(x => x.IsCorrect).ToArray();

     public Action<Control.ControlCollection> NextQuestion;

     private List<Control> CreatedControld = new List<Control>();

     public void SetupNextQuestion(Action<Control.ControlCollection> func)
     {
         NextQuestion = func;
     }
```

Рисунок 4.5 – Код створення нових питань

{get; set;} – автоматична властивість, на в даному прикладі, створює змінні (питання та відповіді) однакового типу. Перевага автоматичних змінних, якщо по суті вони просто звертаються до автоматично створюваної змінної, чому б прямо не звернутися до змінної без автоматичних змінних? Справа в тому, що в будь-який момент часу при необхідності ми можемо розгорнути автоматична змінних в звичайне властивість, додати в нього якусь певну логіку.

```
var question = new SingleAnswerQuiz
            {
                QuestionImage = "/Questions/1.0.png",
                Answers = new Answer[4]
                {
                    new Answer
                    {
                        Id = 1,
                        TextImage = "/Questions/1.1.png",
                        IsCorrect = true
                    },
                    new Answer
                    {
                        Id = 2,
                        TextImage = "/Questions/1.2.png",
                        IsCorrect = false
                    },
                    new Answer
                    {
                        Id = 3,
                        TextImage = "/Questions/1.3.png",
                        IsCorrect = false
                    },
                    new Answer
                    {
                        Id = 4,
                        TextImage = "/Questions/1.4.png",
                        IsCorrect = false
                    }
                }
```

Рисунок 4.6 – Код заповнення питань

Зображено приклад заповнення запитань (рис 4.6 - 4.7) , кожна відповідь є елементом масиву «Answers». В кожному масиві «Answers» треба вказувати кількість відповідей. У кожної відповіді є декількаами параметрами такі як «Id» , «TextImage» , «IsCorrect».

- Id – відповідає за нумерацію
- TextImage – відповідає за текст відповіді
- IsCorrect – відповідає за правильність відповіді

```
public class SingleAnswerQuiz : GeneralQuiz<RadioButton>, IQuiz
    {
        public void Draw(Control.ControlCollection controls)
        {
            DrawQuestion(controls);

            var y = 200;

            Button submit = DrawSubmitButton(controls);

            y = DrawAnswers<RadioButton>(controls, y, submit);

            submit.Location = new Point(100, y + 50);
            CreatedControld.Add(submit);
            controls.Add(submit);
        }

        protected override void EnableSubmitButton(Button submit, Control.ControlCollection controls)
        {
            submit.Enabled = true;
        }

        protected override void SubmitAnswer(Control.ControlCollection controls)
        {
            var choise = controls.OfType<RadioButton>().FirstOrDefault(x => x.Checked)?.Name;
            if (choise == null)
            {
                throw new Exception(TextConstants.Errors.NoAnswerChoosen);
            }
            var choosenAnswer = Answers.FirstOrDefault(x => x.Id == int.Parse(choise.Replace("answer_", "")));

            Score.QuizScore += GetScore(choosenAnswer);
            Score.Label.Text = Score.QuizScore.ToString(CultureInfo.InvariantCulture);

            ShowCorrectAnswers<RadioButton>(controls);
        }

        public override decimal GetScore(params Answer[] choosenAnswer)
        {
            if (choosenAnswer.Length == 1 && choosenAnswer.First().IsCorrect)
            {
                return 1;
            }

            return 0;
        }
    }
}
```

Рисунок 4.7 – Код питань з однією правильною відповіддю

На рисунку 2.6 зображено код батьківського класу запитань з однією або декількома правильними відповідями.

У випадку якщо користувачу із-за помилки програми (бага) вдасться натиснути кнопку «Відповісти» до того як вибрана вірна кількість відповідей є перевірка і користувач побачить повідомлення «Виберіть більше відповідей» або «Ви вибрали забагато відповідей».

Система балів

Правильна відповідь на запитання це 1 бал, у питаннях с декількома правильними відповідями 1/n , де n – це кількість правильних відповідей, наприклад якщо у питанні 4 правильних відповіді і користувач відповів правильно на 3 з них , то він отримає 0.75 бала.

Код відповідаючий за нарахування балів в питаннях з однією правильною відповіддю:

```
public override decimal GetScore(params Answer[] choosenAnswer)
    {
       if (choosenAnswer.Length == 1 && choosenAnswer.First().IsCorrect)
       {
          return 1;
       }
      }
```

Код відповідаючий за нарахування балів в питаннях з декількома правильними відповідями:

```
public override decimal GetScore(params Answer[] choosenAnswer)
        {
            if (choosenAnswer.Length == CorrectAnswers.Length)
            {
                var score = (decimal)choosenAnswer.Where(x => x.IsCorrect).Count() / CorrectAnswers.Length;
                return score;
            }
          }
```

4.3 Опис програми

При запуску програми користувач має змогу бачити початковий екран, де він бачить назву тренажеру та кнопку «Почати» при натисненні на яку розпочинається тестування. (рис 4.8)

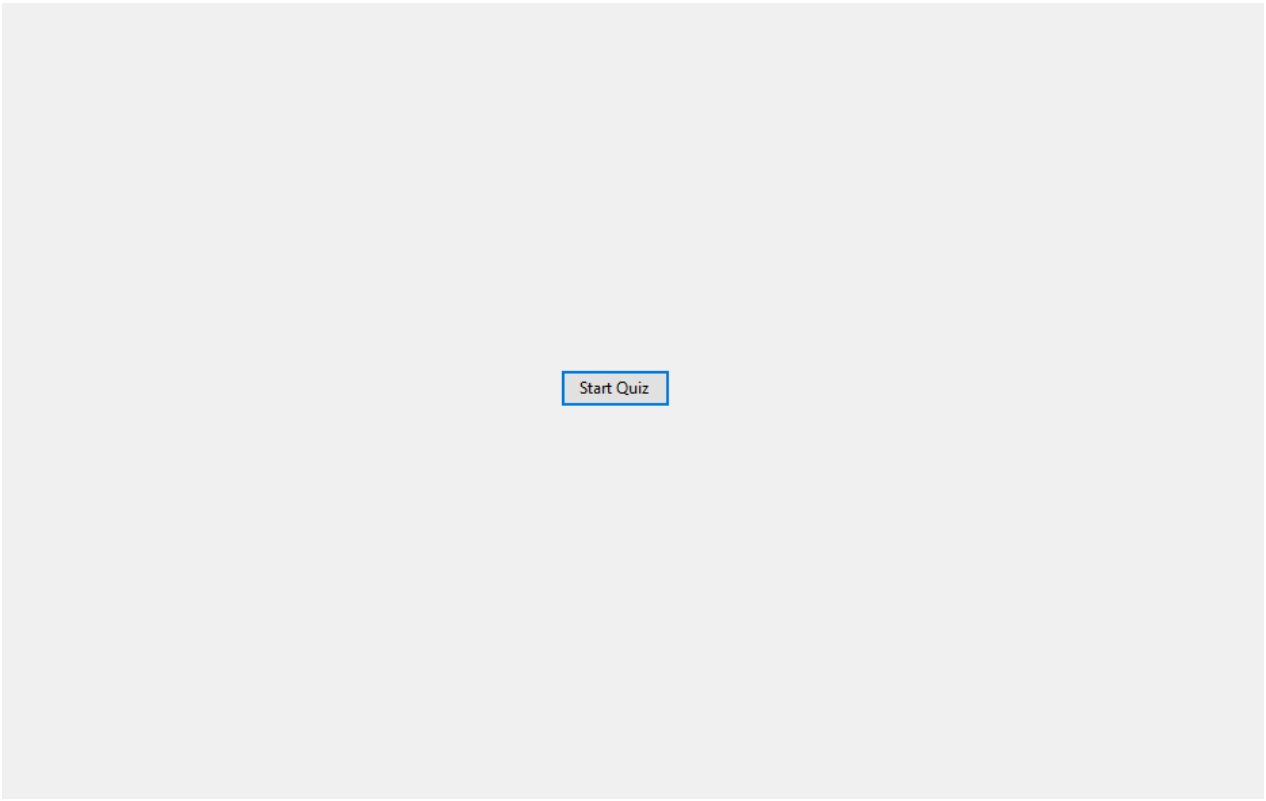

Рисунок 4.8 – Початкове меню тренажера

У тренажері за допомогою тестів перевіряється, чи має користувач відповідні знання та навички. Приклад першого питання , користувач вибравши відповідь на це питання та натиснув кнопку «Відповісти» (рис 4.9) .

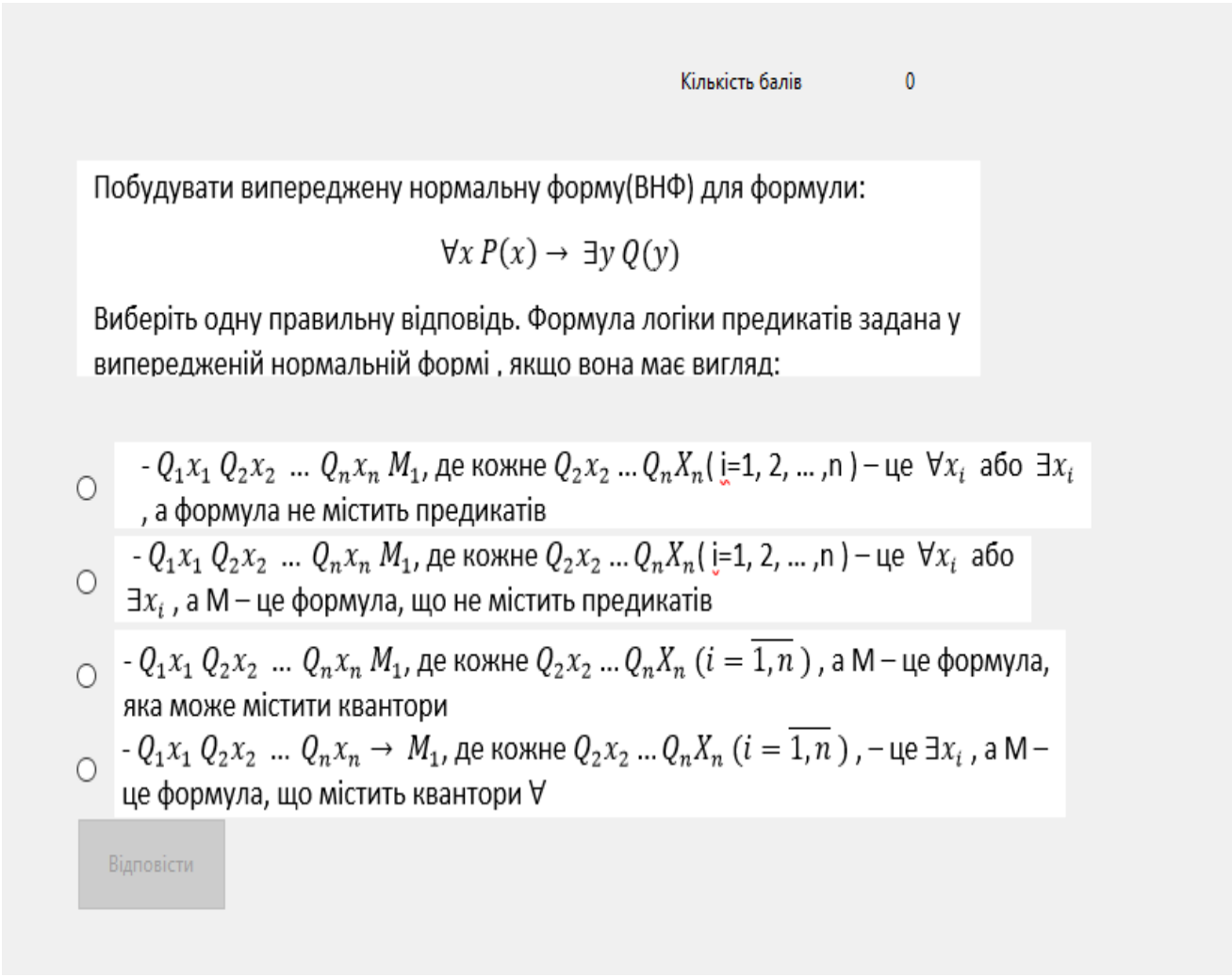

Рисунок 4.9 – Приклад питання з однією правильною відповіддю

Доки користувач не вибрав відповідь кнопка «Відповісти» не працюватиме Рисунок 4.10

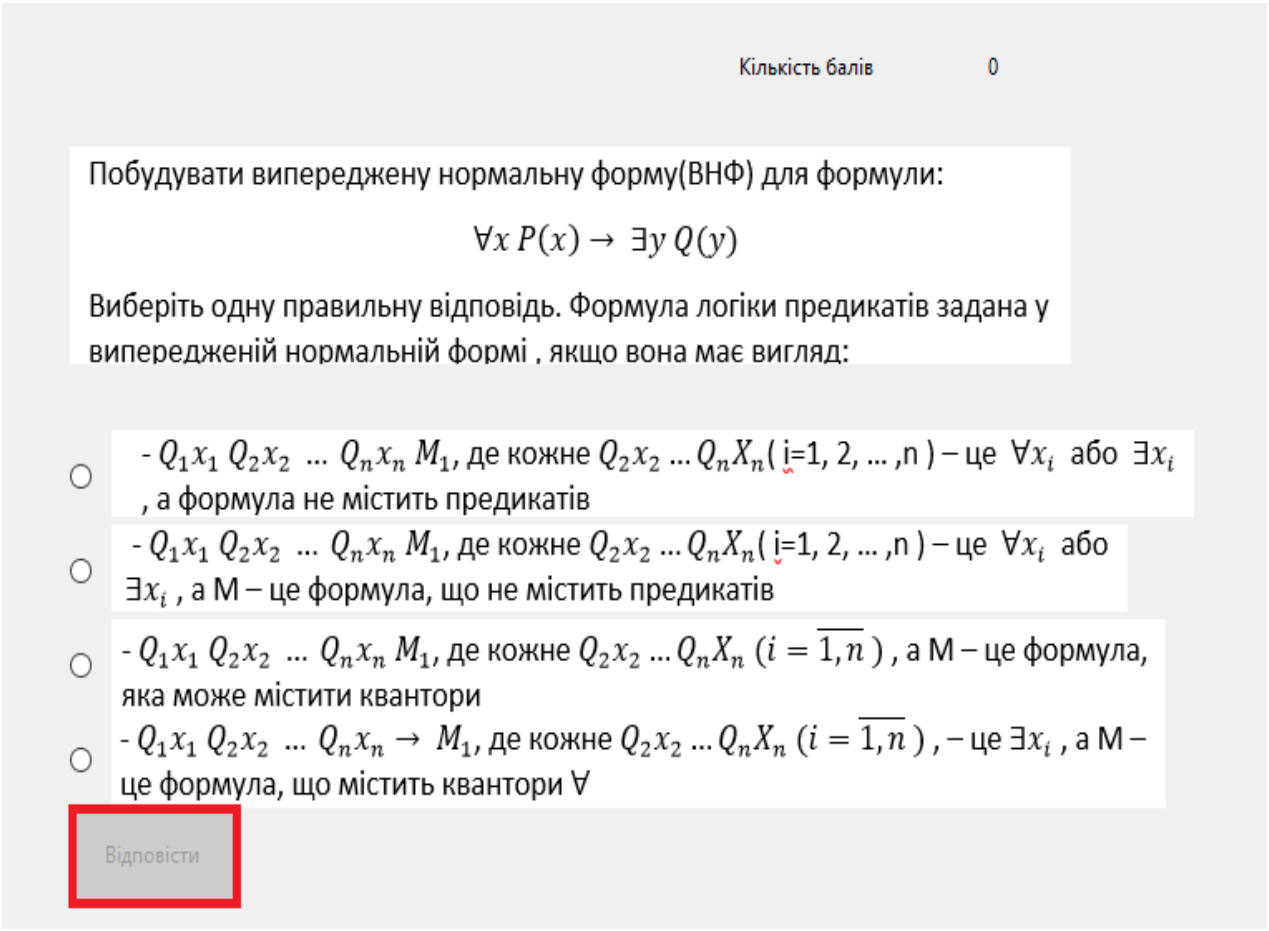

Рисунок 4.10 – Приклад роботи тренажера коли не вибрано відповіді

Після відповіді користувач бачить правильну відповідь (рис 4.11):

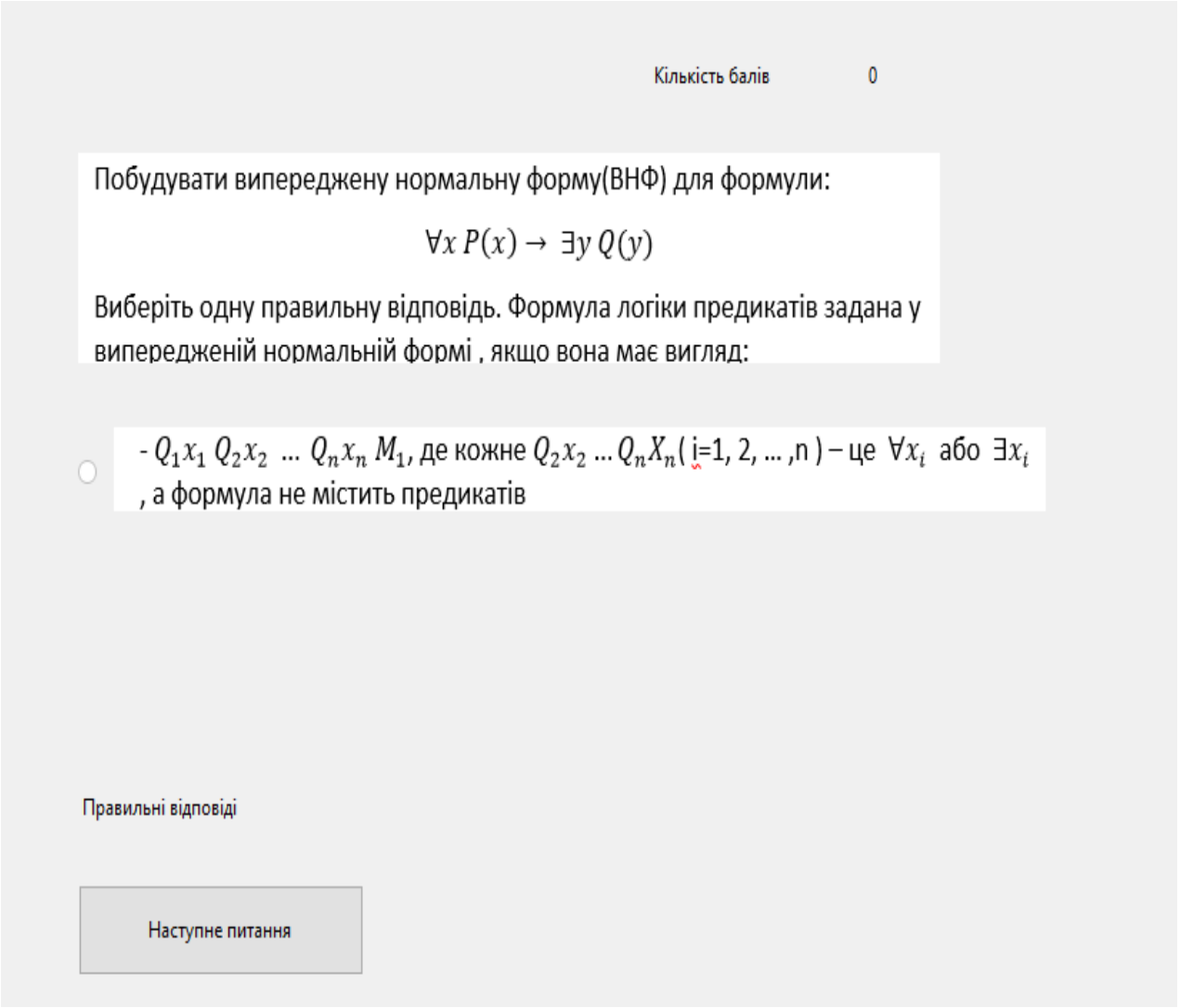

Рисунок 4.11 – Вивід правильної відповіді

Щоб перейти до наступного питання користувач повинен натиснути «Наступне питання»

Це приклад питання з однією правильною відповіддю , у тренажері також є питання с декількома правильними відповідями (Рис 4.12).

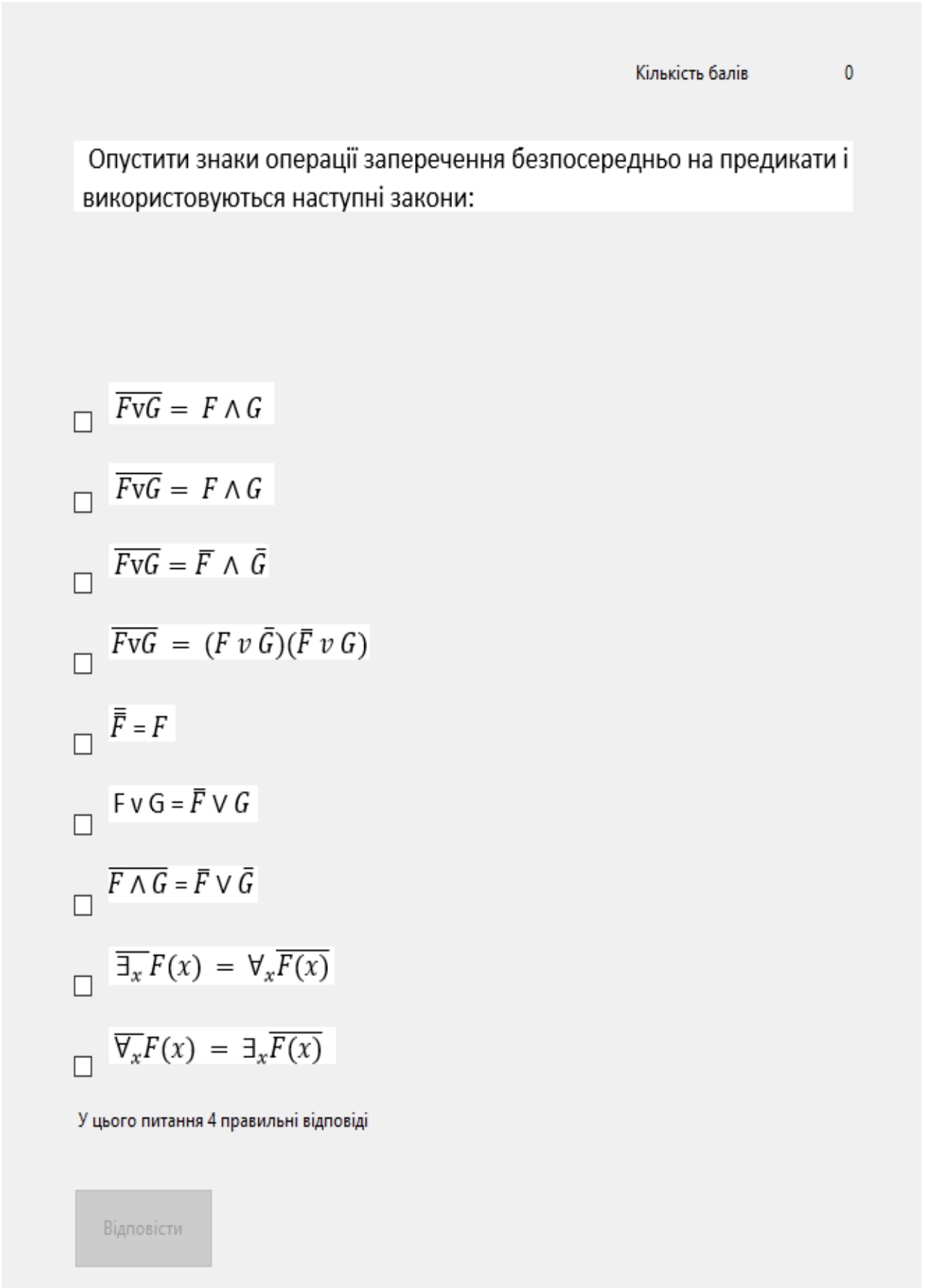

Рисунок 4.12 – Приклад питання з декількома правильними відповідями

Підказкою для користувача є повідомлення про кількість правильних відповідей(Рис 4.13). :

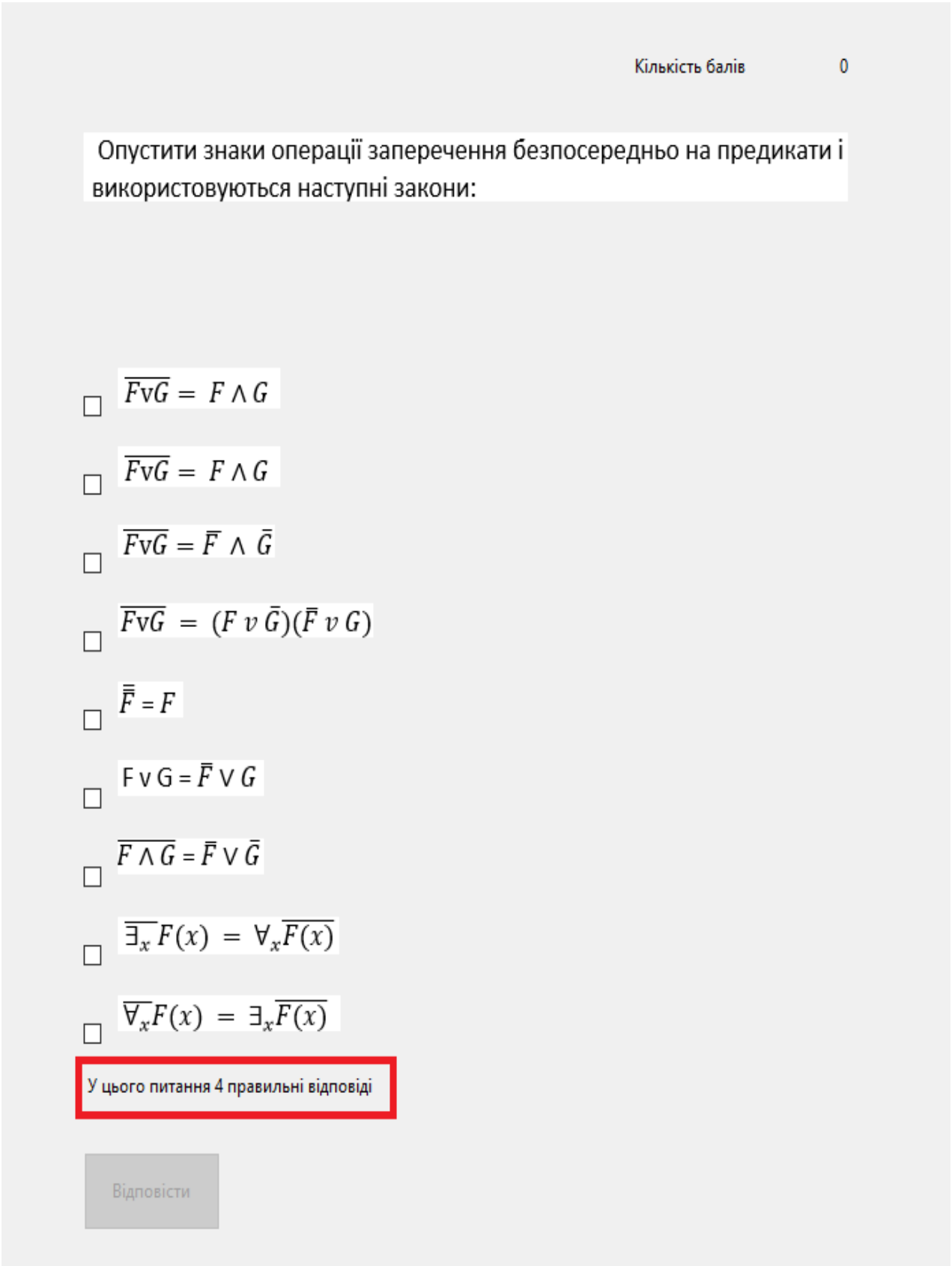

Рисунок 4.13 – Підказка про кількість правильних відповідей

Якщо користувач вибере більше або менше відповідей , він не зможе натиснути кнопку «Відповісти». Щоб перейти до правильної відповіді і до наступного питання. (Рис 4.14 - 4.16).

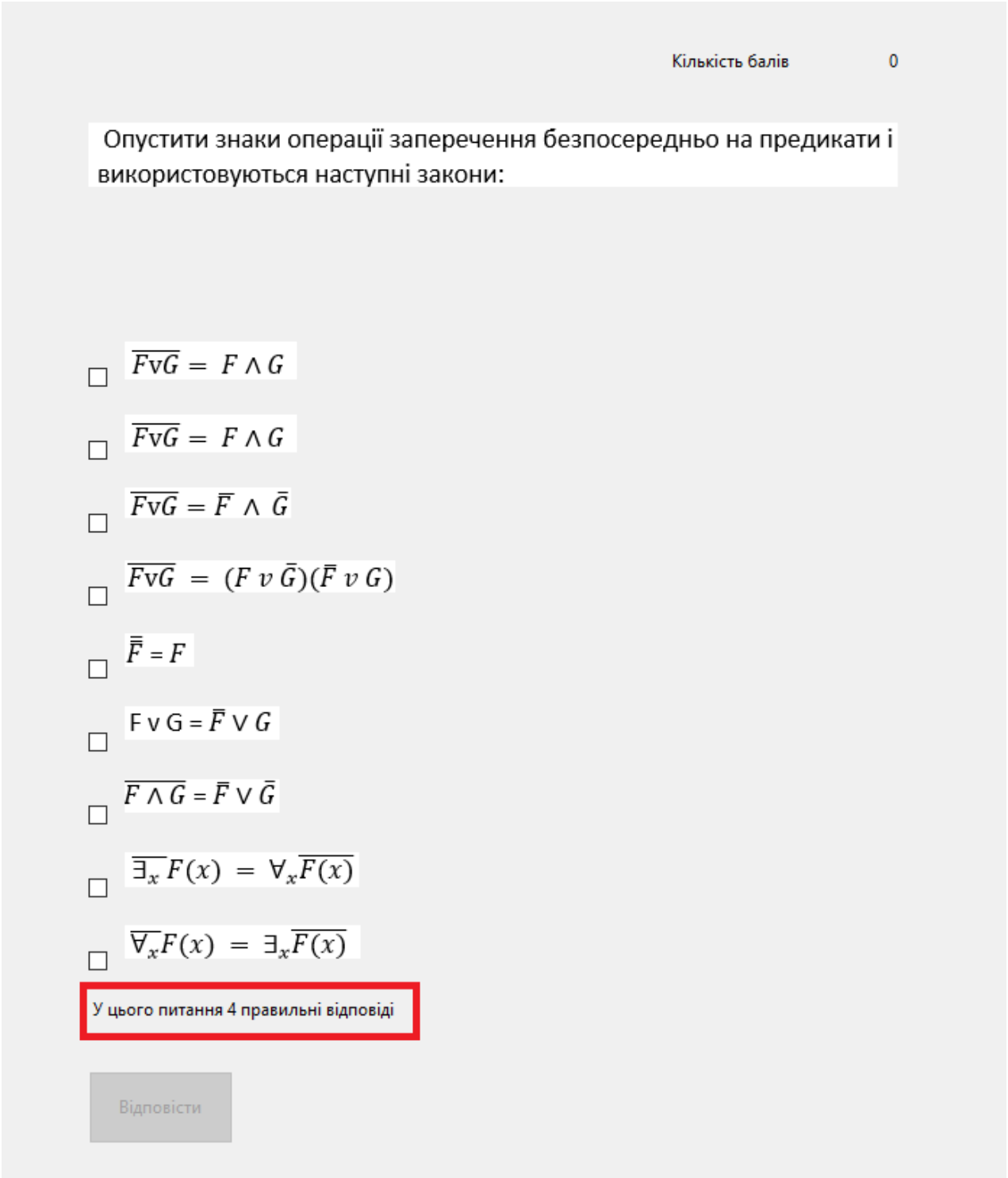

Рисунок 4.14 – Приклад роботи програми коли вибрано забагато відповідей

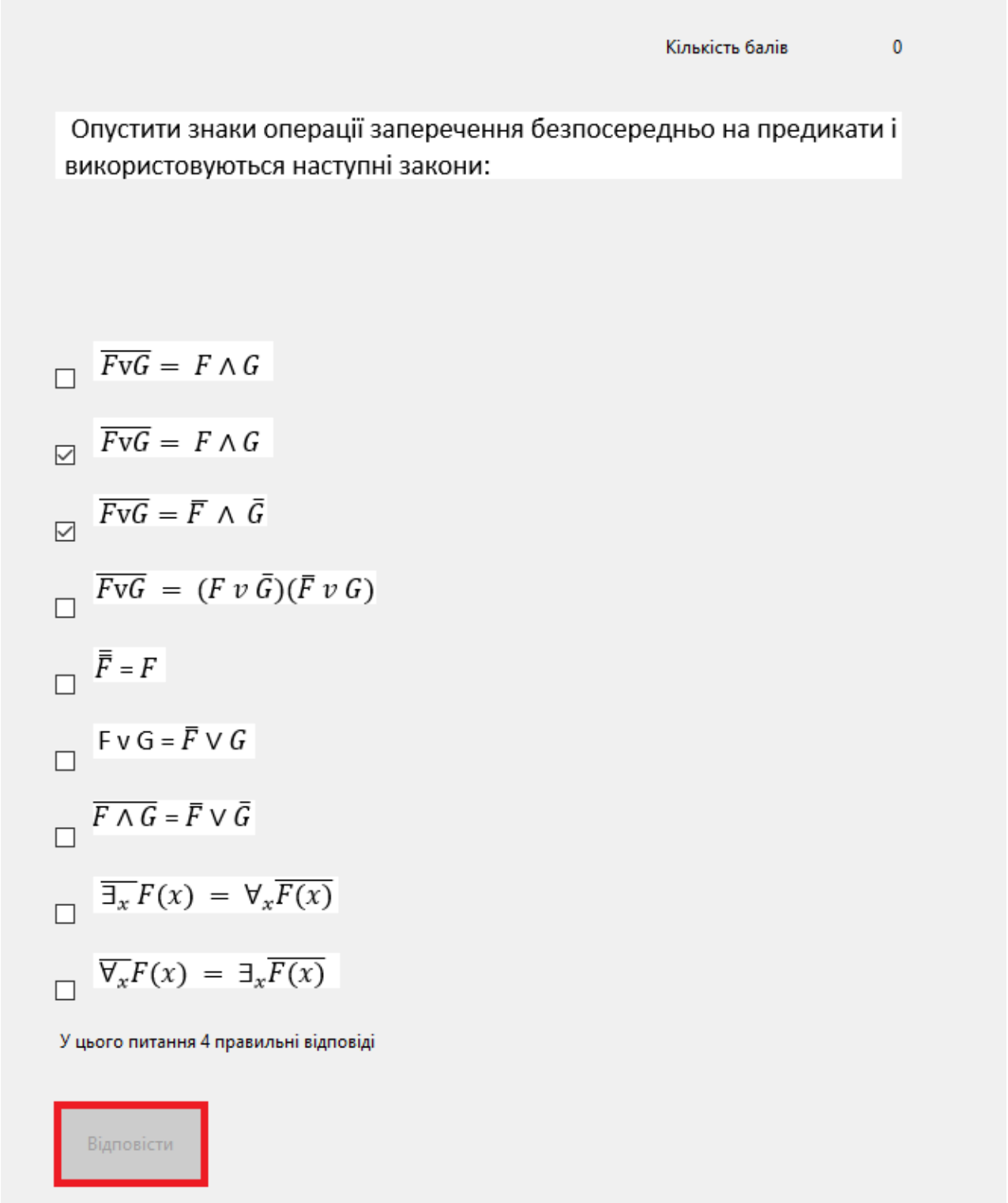

Рисунок 4.15 – Приклад роботи програми коли вибрано замало відповідей

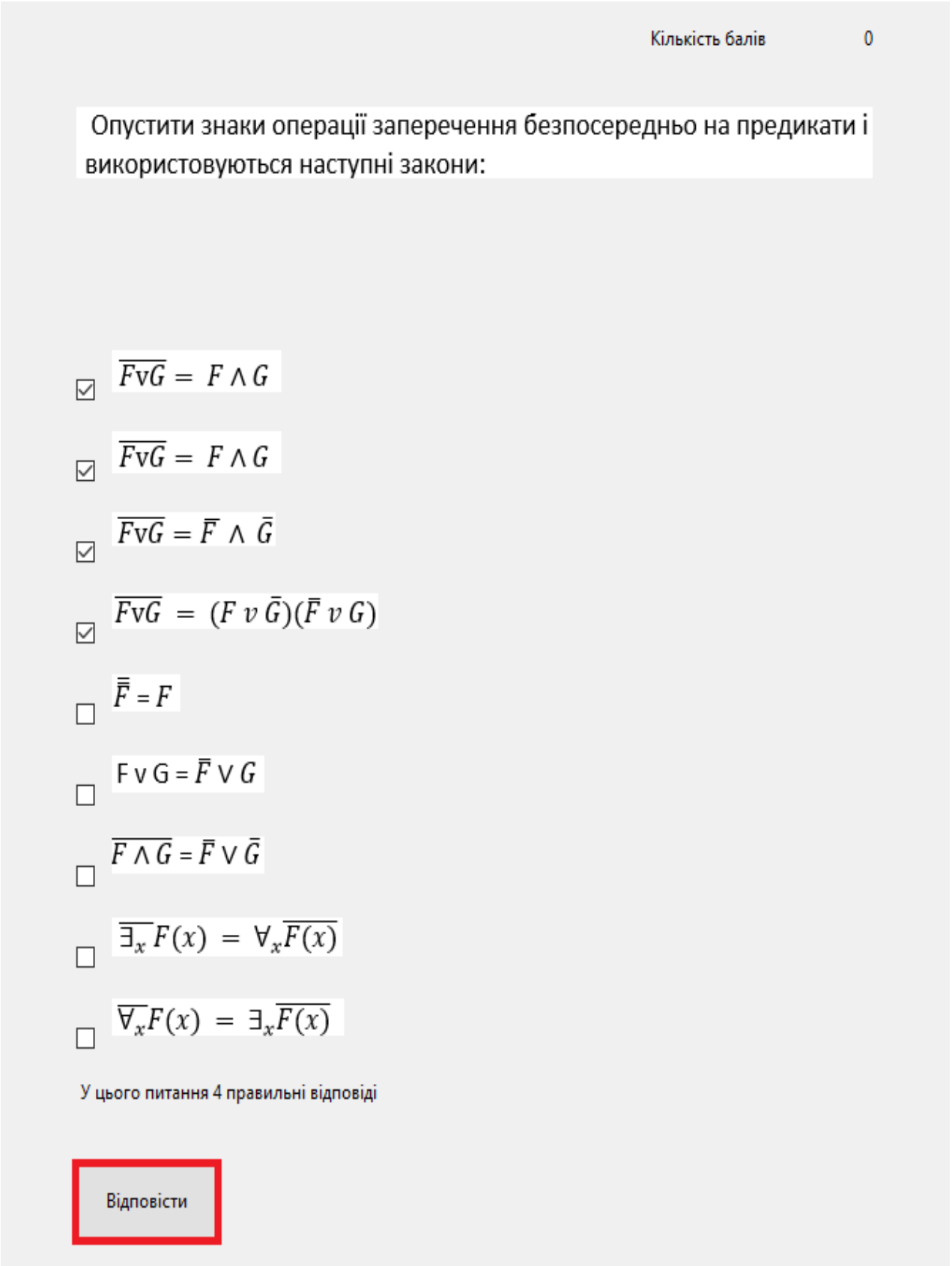

Рисунок 4.16 – Приклад роботи програми коли вибрано правильна кількість відповідей

У тренажері є питання з випадаючими списками (рис 4.17):

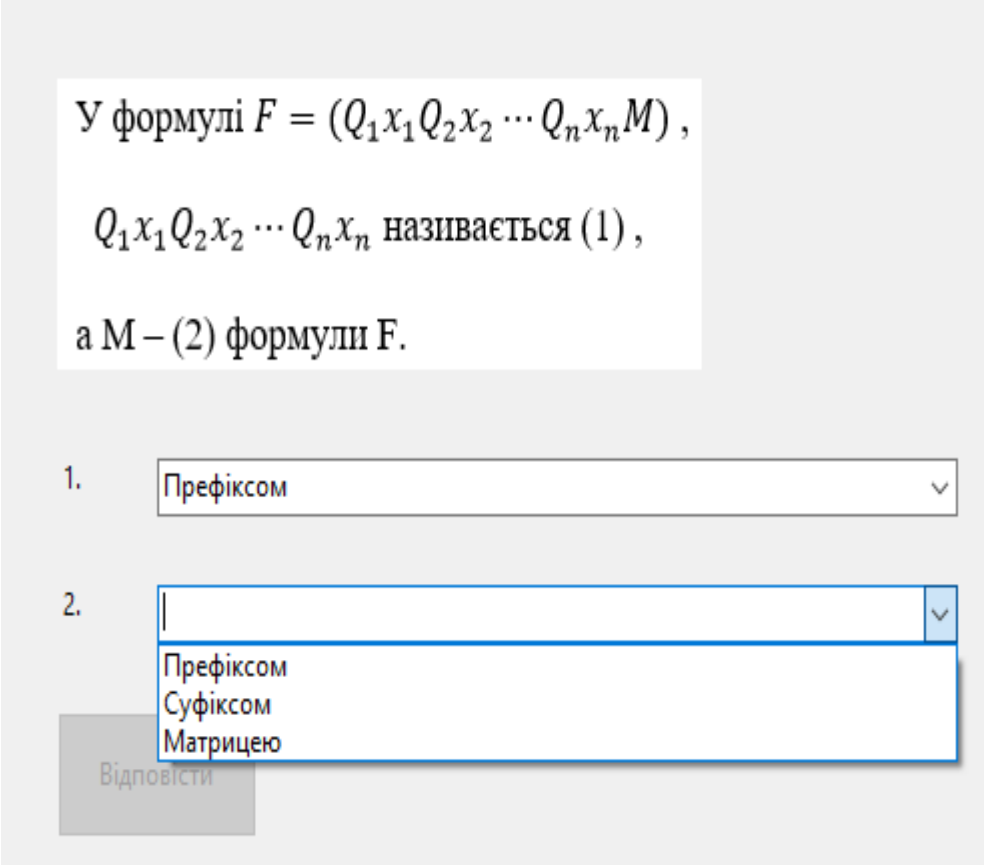

Рисунок 4.16 – Приклад питання з випадаючими списками

Також я к і з попередніми питаннями, якщо не вибрана правильна відповідь користувач не зможе натиснути кнопку відповісти.

Система балів

Користувач може відслідковувати кількість правильних відповідей (рис 4.18 – 4.19)

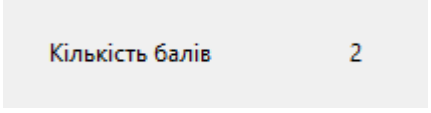

Рисунок 4.18 – Кількість отриманих балів

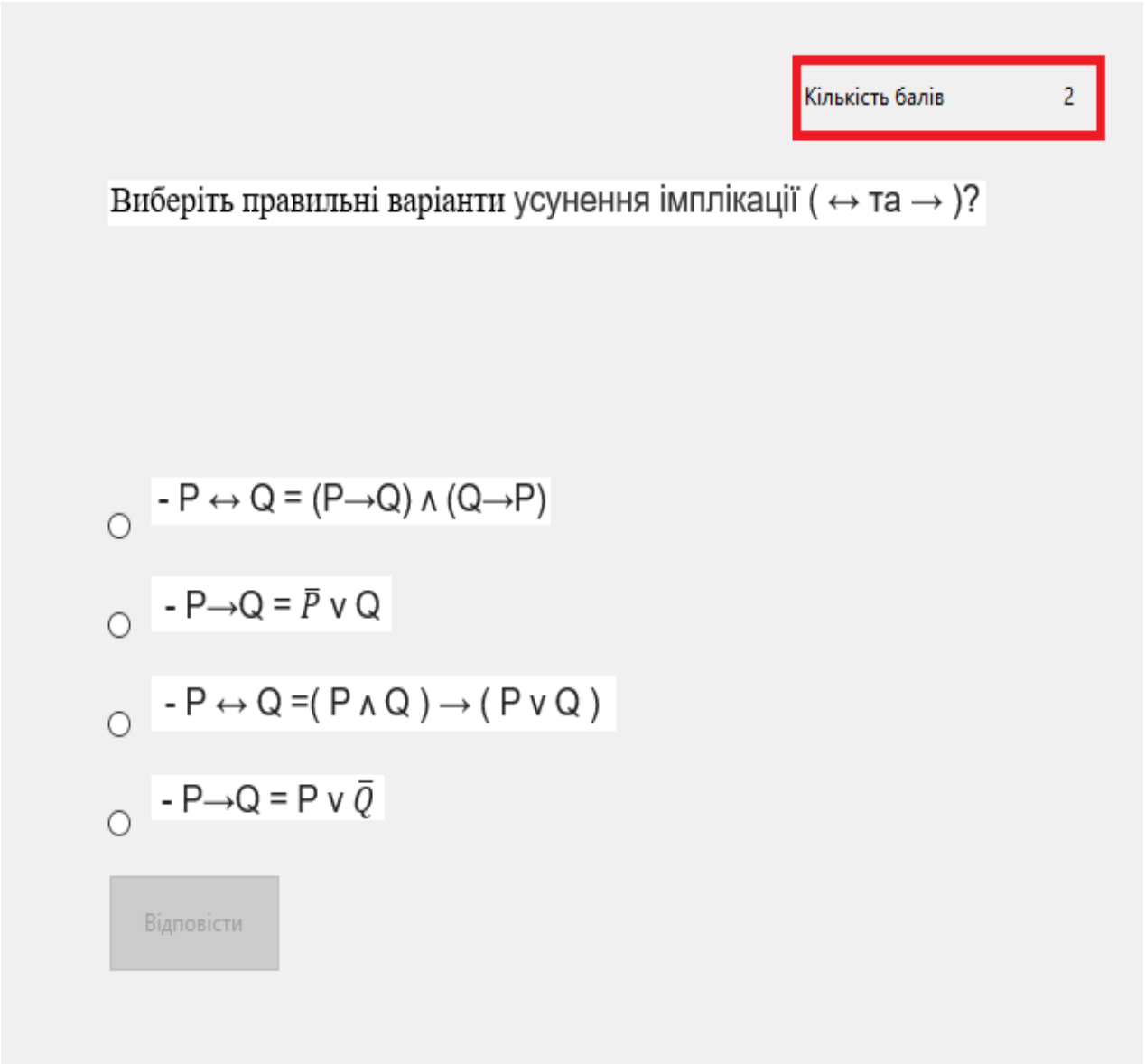

Рисунок 4.19 – Розташування у тренажері кількості балів

Коли користувач відповів на всі питання, з’являється спливаюче вікно з загальною кількістю балів(рис 4.20):

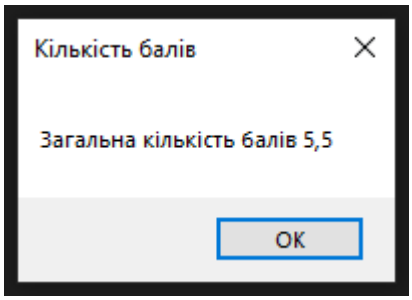

Рисунок 4.20 – Приклад вікна з кількістю балів в кінці роботи тренажера

ВИСНОВОК

Під час виконання бакалаврської роботи були закріпленні теоретичні знання з теми «Випередженої нормальної форми в логіці предикатів» ,які були надані у лекційному матеріалі , було оглянуто попередні близькі до теми роботи, також був розроблений тренажер до теми дистанційного навчального курсу «Математична логіка».

Основні результати роботи:

1)Виконаний інформаційний огляд попередніх робіт;

2)Виділені позитивні і негативні аспекти оглянутих робіт;

3)Розроблено завдання до тренажера;

4)Розроблено алгоритм роботи тренажера до теми «Випереджена нормальна форма в логіці предикатів» дистанційного навчального курсу «Математична логіка»;

5)Створено блок-схеми для програмування.

6)Розглянуто програмні засоби для реалізації завдання роботи;

7)Розробленно тренажер до теми «Випереджена нормальна форма в логіці предикатів» дистанційного навчального курсу «Математична логіка».

8)Було виконано тестування тренажеру.

СПИСОК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.Математична логіка , лекція «Випереджена нормальна форма» Головний центр дистанционного начання. [Електронний ресурс]. –Режим доступу http://www2.el.puet.edu.ua/st/mod/page/view.php?id=119624

2. Об’єкно орієнтоване програмування, [Електронний ресурс].– Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Object-oriented_programming

3.Microsoft [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/csharp/

4. Об’єкно орієнтоване програмування в C#, [Електронний ресурс].– Режим доступу до ресурсу: https://itvdn.com/ru/blog/article/oop-csharp-basic

5. Алієв Ф. П. Тренажер з теми «Матриці і визначники» / Ф. П. Алієв //Парфьонова Т.О. Алгебра і геометрія (Частина 1) 2020н.р. :[дистанційний курс]/Т.О. Парфьонова // Головний центр дистанційного навчання , Полтава: ПУЕТ - Режим доступу – http://www2.el.puet.edu.ua/st/course/view.php?id=792

6. Прімов Х. Н. Тренажер з теми «Метод Крамера» / Х. Н. Прімов //Парфьонова Т.О. Алгебра і геометрія (Частина 1) 2020н.р. :[дистанційний курс]/Т.О. Парфьонова // Головний центр дистанційного навчання , Полтава: ПУЕТ - Режим доступу – http://www2.el.puet.edu.ua/st/course/view.php?id=792

7. Основи математичної логіки, [Електронний ресурс].– Режим доступу до ресурсу: http://www.cyb.univ.kiev.ua/library/training-materials/discrete-mathematics/foundations-of-mathematical-logic.pdf

8. Основи: логіка т аметоди доведення, множини [Електронний ресурс].– Режим доступу до ресурсу: http://itosvita.ucoz.ua/logo/pic_VUZ/chap/Diskretnaya_matematica_01.pdf

9. Випереджені нормальні форми і логічний висновок у логіці предикатів [Електронний ресурс]. – Режим доступу до ресурсу:https://life-prog.ru/ukr/1_3298_viperedzheni-normalni-formi-i-logichniy-visnovok-u-logitsi-predikativ.html

10. Випереджені нормальні форми і логічний висновок у логіці предикатів [Електронний ресурс]. – Режим доступу до ресурсу: https://metanit.com/sharp/patterns/