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
(підпис)
«_____»_____2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО ДИПЛОМНОЇ РОБОТИ

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ТРЕНАЖЕРА З ТЕМИ «МЕТОД РЕКУРСИВНОГО СПУСКУ ПРИ СИНТАКСИЧНОМУ АНАЛІЗІ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ТЕОРІЯ ПРОГРАМУВАННЯ»

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Нечай Валентин Володимирович
_____ «___»_____2021р.
(підпис)
Науковий керівник к.ф.-м.н., доц. Черненко Оксана Олексіївна
_____ «___»_____2021р.
(підпис)

ПОЛТАВА 2021р.

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

ЗАТВЕРДЖУЮ
Завідувач кафедри_____О.О. Ємець
(підпис)
«_1_» вересня 2021 р.

ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ГРАФІК
ВИКОНАННЯ ДИПЛОМНОЇ РОБОТИ

Студент(ка) спеціальності 122 «Комп'ютерні науки»
Прізвище, ім'я, по батькові Нечай Валентин Володимирович
1. **Тема «Програмне забезпечення для тренажера з теми «Метод рекурсивного спуску при синтаксичному аналізі» дистанційного навчального курсу «Теорія програмування»**
затверджена наказом ректора № ___-Н від «___» _____ 2021 р.
Термін подання студентом дипломної роботи _____ «___»_____ 2021 р.

2.Вихідні дані до дипломної роботи: методичні матеріали за темою роботи; матеріали дистанційного навчального курсу «Теорія програмування» та стандарти.

3. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

ВСТУП
1 ПОСТАНОВКА ЗАДАЧІ
2 ІНФОРМАЦІЙНИЙ ОГЛЯД
3 ТЕОРЕТИЧНА ЧАСТИНА
  3.1 Загальні відомості
  3.2 Алгоритм роботи тренажера
  3.3 Блок-схема програми-тренажера
4 ПРАКТИЧНА ЧАСТИНА
  4.1 Опис створення програмного забезпечення
  4.2 Інструкція по використанню програмного забезпечення
ВИСНОВКИ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТОК А

4. Перелік графічного матеріалу (з точним визначенням кількості блок-схем, іншого графічного матеріалу)
Блок-схеми алгоритму, рисунки(в необхідній кількості)

5. Консультанти розділів дипломної роботи

Розділ	Прізвище, ініціали, посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Вступ	Черненко О.О.		
1. Постановка задачі	Черненко О.О.		
2. Інформаційний огляд	Черненко О.О.		
3. Теоретична частина	Черненко О.О.		
4. Практична частина	Черненко О.О.		

6. Календарний графік виконання дипломної роботи

Зміст роботи	Термін виконання	Фактичне виконання
1. Вступ		
2. Вивчення методичних рекомендацій та стандартів та звіт керівнику		
3. Постановка задачі		
4. Інформаційний огляд джерел бібліотек та інтернету		
5. Теоретична частина		
6. Практична частина		
7. Закінчення оформлення		
8. Доповідь студента на кафедрі		
9. Доробка (за необхідністю), рецензування		

Дата видачі завдання «1» вересня 2021 р.

Студент(ка) _____ Нечай Валентин Володимирович
(підпис)

Науковий керівник _____ к.ф.-м.н., доц. Черненко О.О.
(підпис)

Результати захисту дипломної 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: 46 стор., в т.ч. основна частина 42 стор., джерел -14.

Предмет розробки – тренажер з теми «Метод рекурсивного спуску при синтаксичному аналізі».

Мета роботи – створення програмного забезпечення у вигляді тренажеру з теми «Метод рекурсивного спуску при синтаксичному аналізі» для дистанційного навчального курсу «Теорія програмування».

Методи, які були використані для розв'язування задачі –

Програмне забезпечення розроблено за допомогою MS Visual Studio 2019 та платформи Unity 2020.

Під час роботи з тренажером користувач отримує довідкову інформацію для подальшої роботи та практичні завдання у вигляді тестів.

Реалізовано перевірку на правильність введеної відповіді, підказку в повідомленні про неправильну відповідь та можливість повторити роботу після завершення роботи з тренажером.

Ключові слова: ТРЕНАЖЕР, МЕТОД РЕКУРСИВНОГО СПУСКУ, ТЕОРІЯ ПРОГРАМУВАННЯ.

ЗМІСТ

ВСТУП

В новітньому світі не завжди є час на повноційне аудиторне навчання. Саме тому в останній час було створено багато інтернет ресурсів чи програм-вчителів для навчання певним темам. Такі програми відрізняються гнучкістю навчального процесу, дозволяючи навчатися у будь-який час у будь-якому місці, вимагаючи лише наявність електронного пристрою для показу інформації. Для деяких програм навіть не потрібен постійний доступ до мережі інтернет.

Мета роботи – створення тренажеру з теми «Метод рекурсивного спуску при синтаксичному аналізі» для систем дистанційного навчання.

Об’єкт роботи – процес дистанційного навчання студентів.

Предмет роботи – тренажер з теми «Метод рекурсивного спуску при синтаксичному аналізі».

Методи роботи – програмне забезпечення розроблено за допомогою MS Visual Studio 2019 та платформи Unity 2020.

Структура пояснювальної записки до дипломної роботи:

- титульний аркуш;
- завдання до дипломної роботи;
- реферат, що містить предмет, мету, методи, анотацію результатів ключові слова, словосполучення;
- зміст;
- вступ;
- теоретична частина;
- практична частина;
- висновки;
- список використаних джерел;

Обсяг пояснювальної записки: 46 стор., в т.ч. основна частина 42 стор., джерел -14.

1 ПОСТАНОВКА ЗАДАЧІ

Головною метою дипломної роботи є створення тренажеру для систем дистанційного навчання з теми «Метод рекурсивного спуску при синтаксичному аналізі».

Дане програмне забезпечення буде створено у MS Visual Studio 2019, за допомогою платформи Unity 2020 на мові програмування C#.

Основними вимогами до тренажеру є:

1. Можливість навчання без доступу до мережі інтернет;
2. Наявність довідкового матеріалу з теми;
3. Наявність практичних завдань у вигляді тестів;
4. Наявність перевірки введеної відповіді та показ відповідного повідомлення;
5. Зручний та зрозумілий інтерфейс;
6. Можливість повторити роботу з тренажером через відповідну кнопку у кінці навчальних матеріалів.

Основними вимогами до бакалаврської роботи є:

1. Пошук теоретичної інформації;
2. Пошук та створення завдань для тренажеру;
3. Розробка алгоритму роботи тренажера;
4. Розробка блок-схем для роботи з тренажером;
5. Програмна реалізація програми-тренажера;
6. Дотримання вимог оформлення.

2 ІНФОРМАЦІЙНИЙ ОГЛЯД

Для інформаційного огляду було відібрано два тренажери, що також були створені для дистанційного курсу «Теорія програмування», а саме:

- Тренажер з теми «Дерево розбору», автор Алексов, С.В. [2], та
- Тренажер з теми «Контекстовільні граматики», автор Скромінський, М.В. [3]

Обидва тренажери дуже схожі за оформленням та алгоритмом видачі навчальних матеріалів.

2.1 Позитивні та негативні сторони розглянутих тренажерів

Першим було розглянуто тренажер з теми «Дерево розбору», автор Алексов, С.В.

Під час запуску тренажеру користувач отримує інформацію щодо теми та виконавців роботи, а також має можливість обрати з якою частиною працювати одразу, з теорією, чи з практикою через відповідні кнопки.

Під час роботи з практичним матеріалом при виборі неправильної відповіді користувач отримує відповідне повідомлення та правильну відповідь.

Існує можливість обрати складність практичного матеріалу, ця функція заключається у виборі між трьома різними завданнями з різним рівнем складності.

По закінченню роботи з тренажером користувач отримує статистику обраних правильно відповідей.

Головними позитивними сторонами розглянутого тренажеру є:

1. Наявність як практичного так і теоретичного матеріалу;
2. Перевірка введеної відповіді та видача підказки;
3. Можливість обрати рівень складності під час роботи з практичними матеріалами.

Головними негативними сторонами розглянутого тренажеру є:

1. Неможливість перейти від практичного завдання до теоретичного матеріалу з теми без перезавантаження тренажеру;

2. Неможливість повторити роботу з тренажером через вбудований функціонал;

3. Неможливість змінити складність роботи без перезавантаження тренажеру.

Другим було розглянуто тренажер з теми «Контекстовільні граматики», автор Скромінський, М.В.

Під час запуску тренажеру користувач отримує інформацію про тему та виконавців роботи, а також можливість обрати з якою частиною працювати одразу, з теорією, чи з практикою через відповідні кнопки.

Робота з практичною частиною заключається у виборі однієї правильної відповіді у тестах. У разі виборі неправильної відповіді користувач отримує повідомлення про помилку з підказкою в ньому.

По завершенню роботи користувач отримує повідомлення про кінець та можливість пройти тренажер знову.

Головними позитивними сторонами розглянутого тренажеру є:

1. Наявність як практичного так і теоретичного матеріалу;

2. Перевірка введеної відповіді та видача підказки;

3. Можливість повторити роботу з тренажером через відповідну кнопку.

Головними негативними сторонами розглянутого тренажеру є:

1. Неможливість перейти від практичного завдання до теоретичного матеріалу з теми без перезавантаження тренажеру;

2. Не зручний інтерфейс програми.

Дослідивши обрані тренажери було вирішено створити тренажер, що буде повторювати позитивні сторони та уникати негативних з розглянутих тренажерів.

3 ТЕОРЕТИЧНА ЧАСТИНА

3.1 Загальні відомості

Синтаксичний аналіз – це процес зіставлення лінійної послідовності слів (лексем, токенів) мови з формальною граматикою, результатом якого є дерево розбору. Використовується спільно з лексичним аналізом. У процесі синтаксичного аналізу вхідний текст перетворюється в структуру даних (дерево) і добре підходить для подальшої обробки. Розрізняють два типи алгоритмів синтаксичного аналізу: **спадний** – будує дерево розбору від кореня, сворюючи вузли в порядку обходу; **висхідний** – будує дерево розбору для вхідного рядка, починаючи з листя і до кореня.

Лексичний аналіз – процес аналітичного розбору вхідної послідовності символів з метою отримання на виході токену (подібно угрупуванню літер у словах).

Синтаксичний аналіз є найбільш важливою частиною процесу компіляції. У теорії компіляції замість терміна “магазин” частіше використовують термін “стек”, тому в даному розділі ми також будемо використовувати цю назву.

Розглянемо спадний аналіз в його загальному вигляді – методом **рекурсивного спуску**, який може використовувати відкіт, іншими словами, виконувати повторне сканування вхідного потоку.

Приклад 1.

Розглянемо граматику з продукціями $S \to cAd$, $A \to ab \mid a$ та вхідний рядок $w = cad$.

При спадній побудові дерева розбору для цього рядка ми спочатку створюємо дерево, що складається з одного вузла, позначеного як S. Покажчик входу вказує на c, перший символ рядка w. Тепер скористаємося першою продукцією для S щоб одержати дерево, зображене на рис. 3.1.

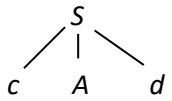


Рисунок 3.1 - Дерево 1 для $w = cad$

Крайній ліворуч лист c відповідає першому символу w, перемістимо покажчик входу до a, другого символу рядка w, і розглянемо наступний лист дерева, позначений A. Тепер можна скористатися для A першою альтернативою й одержати дерево, зображене на рис. 3.2.

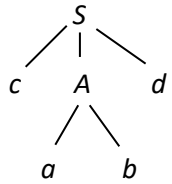


Рисунок 3.2 - Дерево 2 для рядка $w = cad$

Виявлена відповідність зчитаного символа a листу дерева, переходимо до наступного символа d. Однак d не відповідає листу дерева b, а отже, необхідно повернутися до A для того, щоб вибрати нову альтернативу для роботи.

Повертаючись до A, необхідно повернути покажчик у позицію 2, у якій він був, коли ми вперше прийшли до розкладання A. Це означає, що процедура для A повинна зберігати покажчик входу в локальній змінній. При розгляді другої альтернативи для A одержуємо дерево, зображене на рис. 3.3

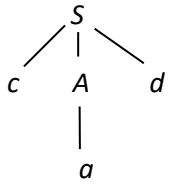


Рисунок 3.3 - Дерево 3 для рядка $w = cad$

Лист a відповідає другому символу w, а лист d – третьому. Оскільки в цей момент побудоване дерево розбору для w, припиняємо роботу і повідомляємо про успішне завершення розбору. [4]

Рекурсивний спуск — алгоритм синтаксичного аналізу, будується на основі взаємно рекурсивних процедур (або не рекурсивних еквівалентів), кожна із яких реалізує одну із продукцій граматики. [5]

3.2 Алгоритм роботи тренажера

Крок 1. При запуску застосунку користувачу надається обрати розширення екрану для комфортної роботи при будь-яких конфігураціях персональних компьютерів.

Крок 2. Користувач переходить до головного меню тренажеру де отримує інформацію про тему роботи.

Крок 3. Користувач переходить до довідкового матеріалу з теми «Метод рекурсивного спуску при синтаксичному аналізі» натиснувши на відповідну кнопку.

Крок 4. Видача довідкового матеріалу:

«Для опису синтаксису мов програмування використовуються КВ граматики, правила яких мають вигляд

A → a, де A є N, a є (T U N) *.

Для різних підкласів КВ-граматик існують досить ефективні алгоритми розбору.

Деякі загальні алгоритми аналізу по КВ-граматиках:

- синтаксичний аналіз з поверненнями (час роботи експоненціально залежить від довжини ланцюжка);

- алгоритм Кока-Янгера-Касамі (для розбору ланцюжків довжини n потрібен час cn^3);

- алгоритм Ерлі (для розбору ланцюжків довжини n потрібен час cn^3).

Ці (і подібні до них за часом роботи) алгоритми практично неприйнятні.

Алгоритми аналізу, які витрачають на обробку вхідного ланцюжка лінійний час, застосовні тільки до деяких підкласів КВ-граматик.»

Крок 5. Отримання практичних завдань у вигляді тестів.

«Завдання 1. Які граматики називають КВ-граматиками?

1. Контексто-вільні граматики

2. Контексто-залежні граматики

3. Граматики з використанням машин Тьюрінга

Правильна відповідь - 1.»

Крок 6. Отримання практичних завдань у вигляді тестів.

«Завдання 2. У випадку роботи з алгоритмом Кока-Янгера-Касамі та Ерлі яка формула для часу розбору?

1. cn^2

2. cn

3. cn^3

Правильна відповідь – 3.»

Крок 7. Отримання практичних завдань у вигляді тестів.

«Завдання 3. Чому розглянуті вище алгоритми практично неприйнятні для більшості КВ-граматик?

1. Багато часу на розбір

2. Використовують лінійний час

3. Застаріли

Правильна відповідь – 2.»

Крок 8. Отримання практичних завдань у вигляді тестів.

«Завдання 4. До якої граматики належить метод рекурсивного спуску?

1. КВ-граматики

2. LR-граматики

3. SP-граматики

Правильна відповідь – 1.»

Крок 9. Видача довідкового матеріалу:

«Застосовність синтаксичних аналізаторів

Кожен метод синтаксичного аналізу заснований на своїй техніці побудови дерева виведення і передбачає свій спосіб побудови по граматиці програми-аналізатора, яка буде здійснювати розбір ланцюжків.

Коректний аналізатор завершує свою роботу для будь-якої вхідний ланцюжка і видає вірну відповідь про належність ланцюжка мови.

Аналізатор некоректний, якщо:

- не розпізнає хоча б один ланцюжок, що належить мові;

- розпізнає хоча б один ланцюжок, мови не належала;
- зациклюється на будь-якої ланцюжку.

Метод аналізу застосуємо до даної граматиці, якщо аналізатор, побудований в Відповідно до цього методу, коректний і будує все можливі висновки ланцюжків в даній граматиці»

Крок 10. Отримання практичних завдань у вигляді тестів.

«Завдання 5. Який аналізатор називається коректним?

1. Видає правильну відповідь для конкретного вхідного ланцюжка
2. Видає правильну відповідь для будь-якого вхідного ланцюжка
3. Видає відповідь на певному кроці

Правильна відповідь – 2.»

Крок 11. Отримання практичних завдань у вигляді тестів.

«Завдання 6. Який аналізатор називається некоректним?

1. Видає правильну відповідь для будь-якого вхідного ланцюжка
2. Не розглядає принаймні один ланцюжок з заданої мови
3. Видає відповідь лише для простих мов

Правильна відповідь – 2.»

Крок 12. Видача довідкового матеріалу:

«Метод рекурсивного спуску (РС-метод)

Нехай дана граматика G_pc = ({a, b, c, _|_}, {S, A, B}, P, S), де

P: L (G) = {$c^n abc^m a\_|\_$ | n, m> = 0}

S → AB_|_

A → a | cA

B → bA

і треба визначити, чи належить ланцюжок caba мови L (G).

Побудуємо висновок цього ланцюжка:

S → AB_|_ → cAB_|_ → caB_|_ → cabA_|_ → caba_|_

Отже, ланцюжок належить мові L (G).

Послідовність застосувань правил виведення еквівалентна побудови дерева розбору методом "зверху вниз", а метод

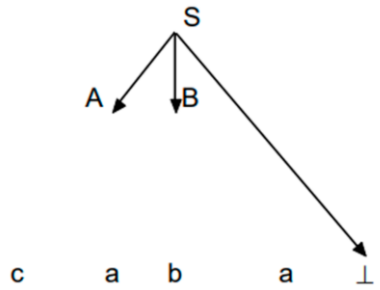
рекурсивного спуску фактично реалізує цей метод аналізу.»

Крок 13. Отримання практичного завдання:

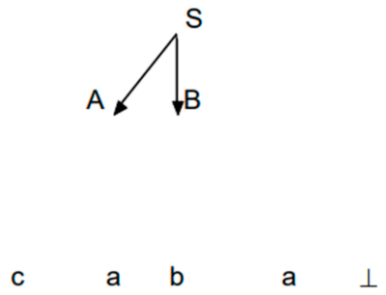
«Побудувати дерево розбору ланцюжка S → AB_|_ → cAB_|_ → caB_|_ → cabA_|_ → caba_|_»

Крок 14. Отримання практичних завдань у вигляді тестів.

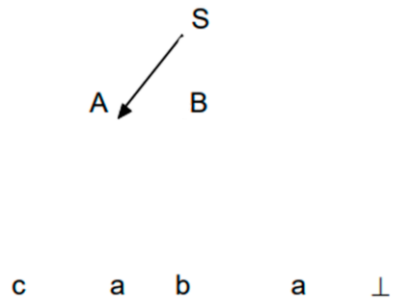
«Завдання 7. Як виглядає перший крок розбору ланцюжка?



1.



2.

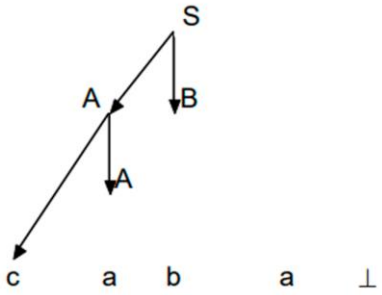


3.

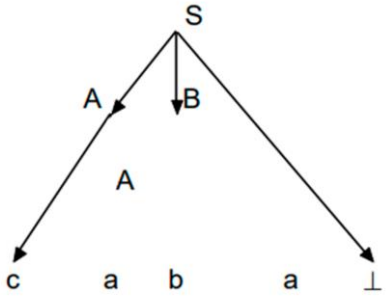
Правильна відповідь – 1.»

Крок 15. Отримання практичних завдань у вигляді тестів.

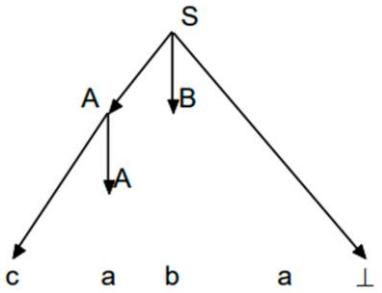
«Завдання 8. Як виглядає другий крок розбору ланцюжка?



1.



2.

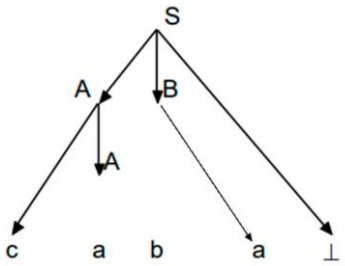


3.

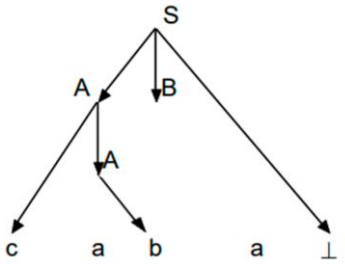
Правильна відповідь – 3.»

Крок 16. Отримання практичних завдань у вигляді тестів.

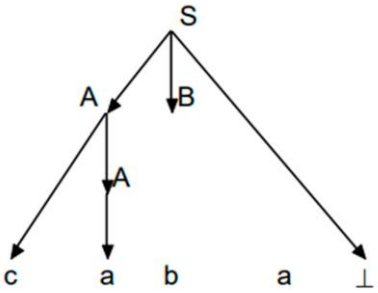
«Завдання 9. Як виглядає третій крок розбору ланцюжка?



1.



2.

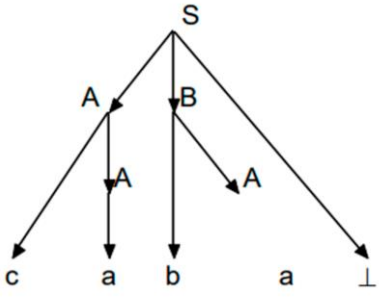


3.

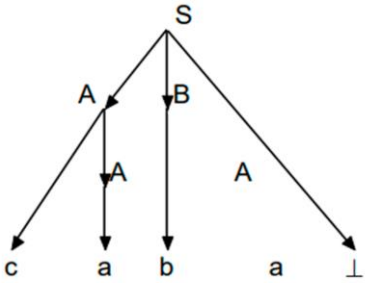
Правильна відповідь – 3.»

Крок 17. Отримання практичних завдань у вигляді тестів.

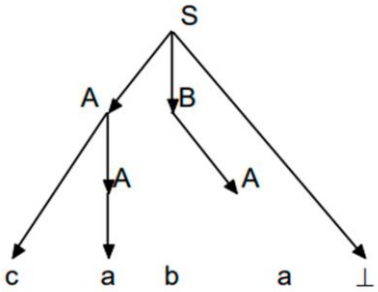
«Завдання 10. Як виглядає четвертий крок розбору ланцюжка?



1.



2.

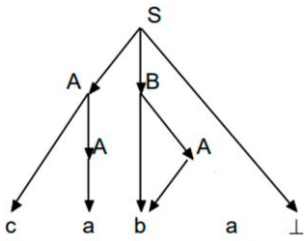


3.

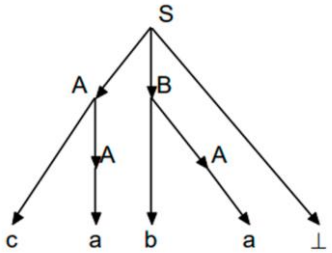
Правильна відповідь – 1.»

Крок 18. Отримання практичних завдань у вигляді тестів.

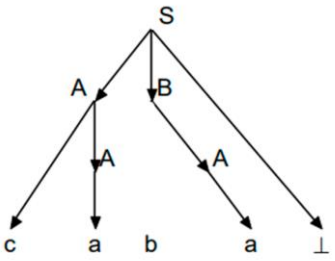
«Завдання 11. Як виглядає останній крок розбору ланцюжка?



1.



2.



3.

Правильна відповідь – 2.»

Крок 19. Вивід повідомлення про успішне завершення роботи та надання доступу до кнопки для повторення роботи.

3.3 Блок-схема програми-тренажера

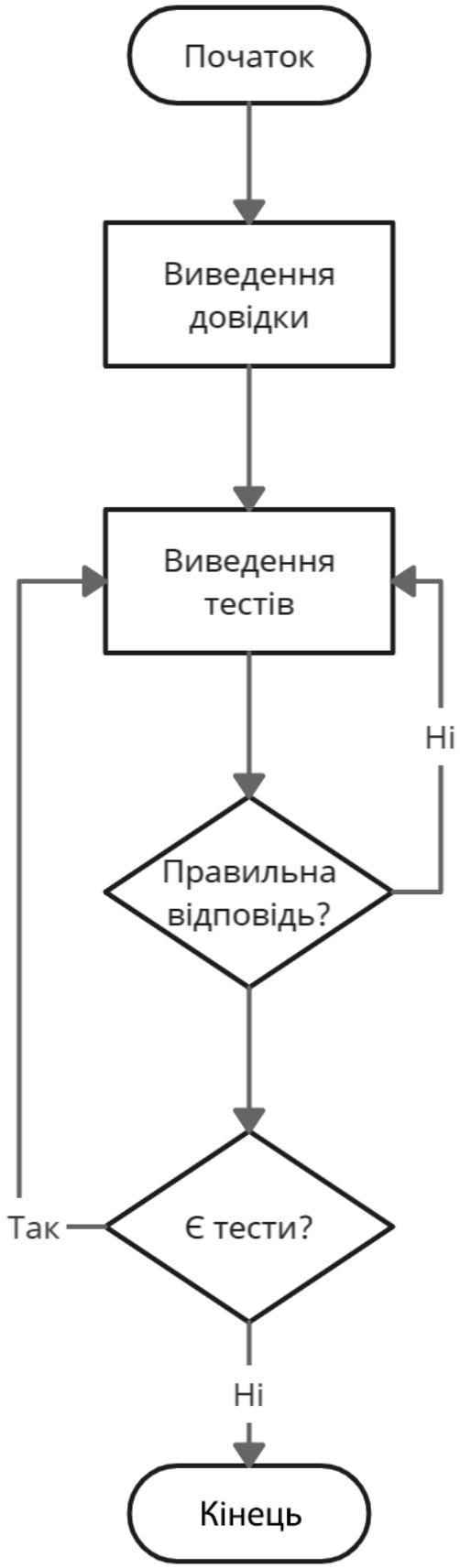


Рисунок 3.4 – Блок-схема програми-тренажера

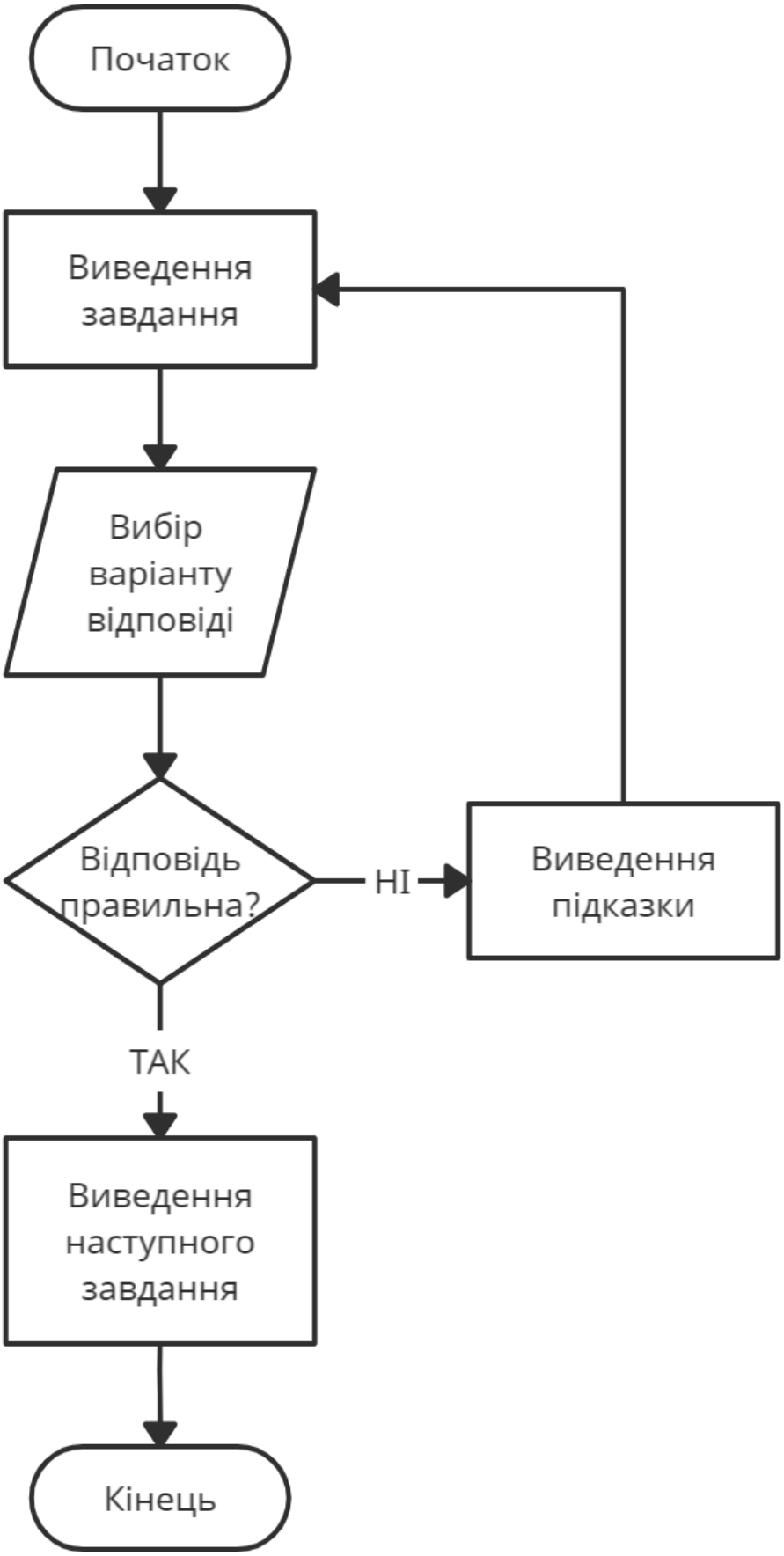


Рисунок 3.5 – Блок-схема роботи з практичними завданнями в тренажері

4 ПРАКТИЧНА ЧАСТИНА

4.1 Опис створення програмного забезпечення

Unity – багатоплатформовий інструмент розробки відеоігор і застосунків, і рушій, на якому вони працюють. Створені за допомогою Unity програми працюють на настільних комп'ютерних системах, мобільних пристроях та гральних консолях у дво- та тривимірній графіці, та на пристроях віртуальної чи доповненої реальності. Застосунки, створені за допомогою Unity, підтримують DirectX та OpenGL.

Unity — це кросплатформовий ігровий рушій. Програма-редактор Unity працює на Windows, macOS і Linux, а сам рушій може запускатися на 25 платормах, а саме iOS, Android, Tizen, Windows, Universal Windows Platform, Mac, Linux, WebGL, PlayStation 4, PlayStation Vita, Xbox One, 3DS, Oculus Rift, Google Cardboard, Steam VR, PlayStation VR, Gear VR, Windows Mixed Reality, Daydream, Android TV, Samsung Smart TV, tvOS, Nintendo Switch, Xbox Series X та Series S, PlayStation 5, Facebook Gameroom, Apple ARKit, Google ARCore, Vuforia, і Magic Leap.

Ігрова логіка пишеться за допомогою мови C#, раніше також була можливість використовувати Boo та JavaScript, але розробники відмовились від їх підтримки.

Редактор Unity має інтерфейс, що складається з різних вікон, які можна розташувати на свій розсуд. Завдяки цьому можна проводити налагодження гри чи застосунка прямо в редакторі. Головні вікна — це оглядач ресурсів проекту, інспектор поточного об'єкта, вікно попереднього перегляду, оглядач сцени та оглядач ієрархії ресурсів.

Проект в Unity поділяється на сцени (рівні) — окремі файли, що містять свої ігрові світи зі своїм набором об'єктів, сценаріїв, і налаштувань. Сцени можуть містити в собі як об'єкти-моделі (ландшафт, персонажі, предмети довкілля тощо), так і порожні ігрові об'єкти — ті, що не мають моделі, проте задають поведінку інших об'єктів (тригери подій, точки збереження прогресу

тощо). Їх дозволяється розташовувати, обертати, масштабувати, застосовувати до них скрипти. В них є назва (в Unity допускається наявність двох і більше об'єктів з однаковими назвами), може бути тег (мітка) і шар, на якому він повинен відображатися. Так, у будь-якого предмета на сцені обов'язково наявний компонент Transform — він зберігає в собі координати місця розташування, повороту і розмірів по всіх трьох осях. У об'єктів з видимою геометрією також за умовчанням присутній компонент Mesh Renderer, що робить модель видимою. Різні моделі можуть об'єднуватися в набори (ассети) для швидкого доступу до них. Наприклад, моделі споруд на спільну тему.

Unity підтримує фізику твердих тіл і тканини, фізику типу Ragdoll (ганчіркова лялька). У редакторі є система успадкування об'єктів; дочірні об'єкти будуть повторювати всі зміни позиції, повороту і масштабу батьківського об'єкта. Скрипти в редакторі прикріплюються до об'єктів у вигляді окремих компонентів.

У 2D іграх Unity переважно використовує спрайти. В 3D іграх Unity здебільшого використовує тривимірні моделі (меші), на які накладаються текстури (зумовлюють вигляд поверхні об'єктів), матеріали (зумовлюють як поверхня реагуватиме на різні фактори) та шейдери (невеликі скрипти, за яким вираховується зміна кольору кожного пікселя згідно заданих параметрів, як-от розсіяння відбитого світла). В обох видах застосовуються системи часток для відображення субстанцій, таких як рідини чи дим.

Unity підтримує стиснення текстур, міпмапінг і різні налаштування роздільності екрана для кожної платформи. [6]

Для створення тренажеру було використано такі функції ігрового двигуна як створення об’єктів (кнопок) та прикріплення до них скріптів для навігації.

Всі елементи (слайди) на яких буде розміщено навчальні матеріали чи тести реалізовано у вигляді різних сцен гри, навігація між якими можлива по натисненні відповідної кнопки.

Для початку роботи потрібно створити певну ієрархію, що буде використовуватися в програмі (див. Рисунок 4.1)

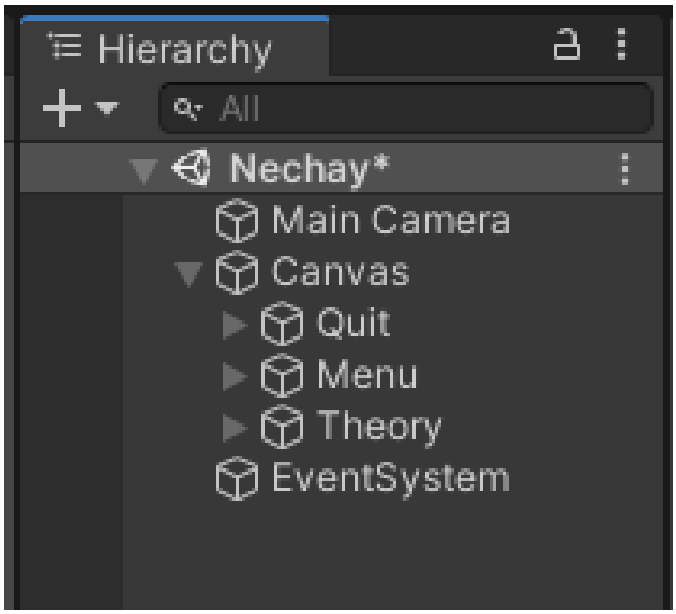


Рисунок 4.1 – Вікно ієрархії в Unity 2020

Кожен елемент може містити в собі як елементи так і скріпти, або і те і те (див. Рисунок 4.2)

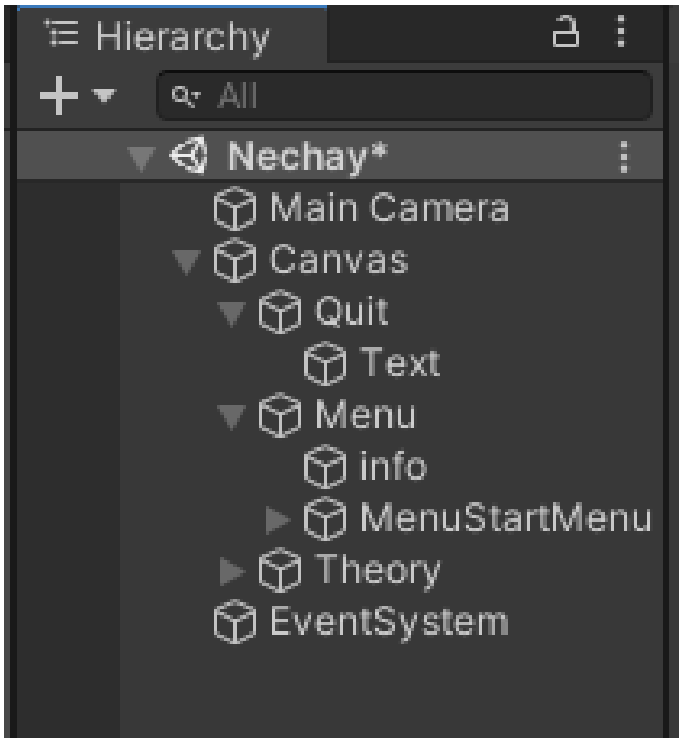


Рисунок 4.2 – Ієрархія дерева в Unity 2020

Прикріплення скріпту відбувається через вбудовані функції Unity 2020. Код для основних скріптів винесено в додатки.

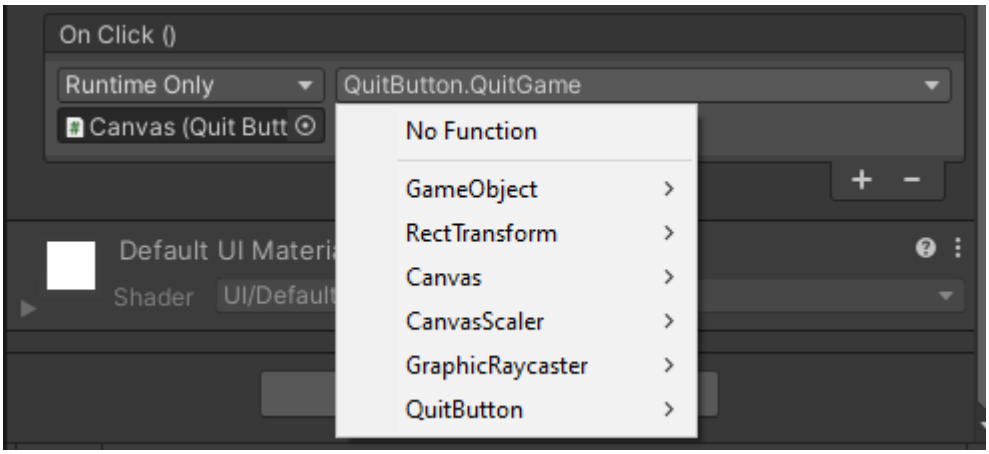


Рисунок 4.3 – Додання скріпту до кнопки виходу

Перехід між елементами (слайдами) тренажеру реалізовано через кнопки Далі, на кожному слайді є доступ до кнопки Вихід.

Вибір налаштувань для тексту у вікні Unity 2020 виглядає наступним чином.

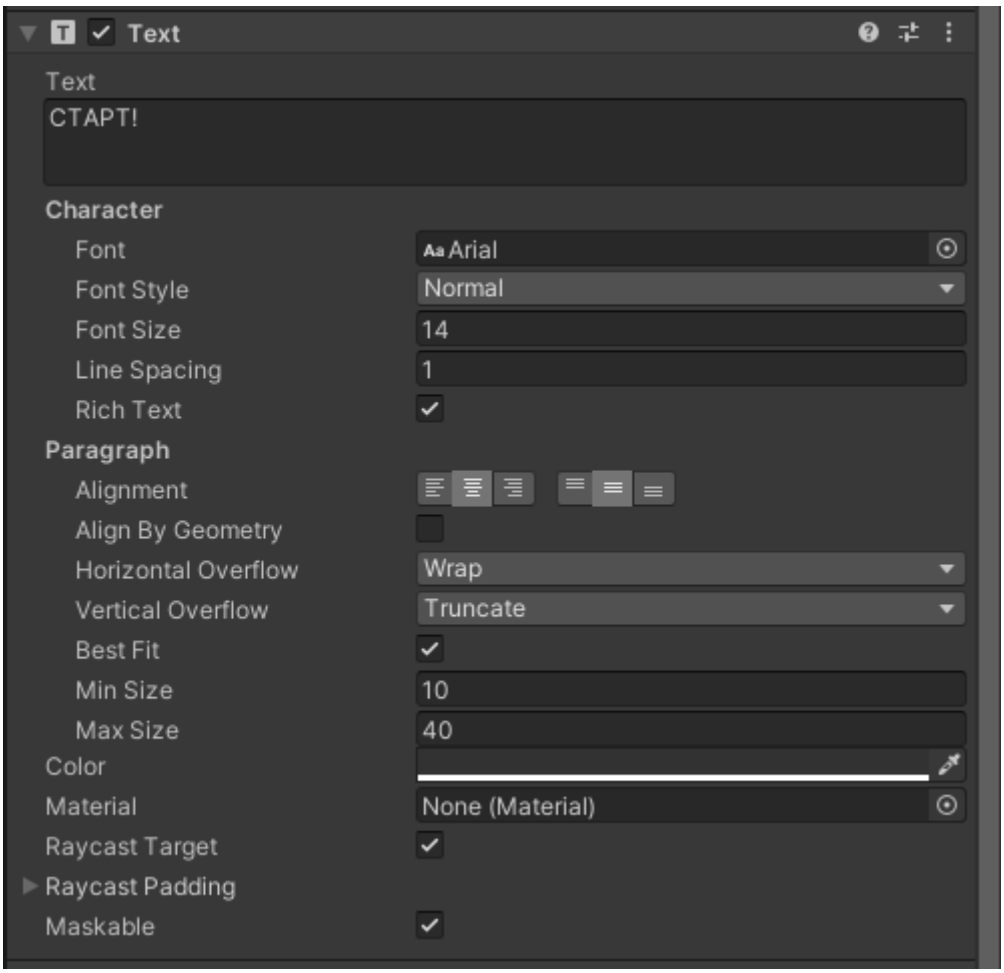


Рисунок 4.4 – Налаштування для тексту

Робота з кнопка в Unity 2020 може бути як можливість застосувати скріпт, що підв’язаний до елементу, що вище в ієрархії, так і як робота з логічними функціями, як закриття однієї сцени (слайду) та відкриття наступної або будь-якої обраної.

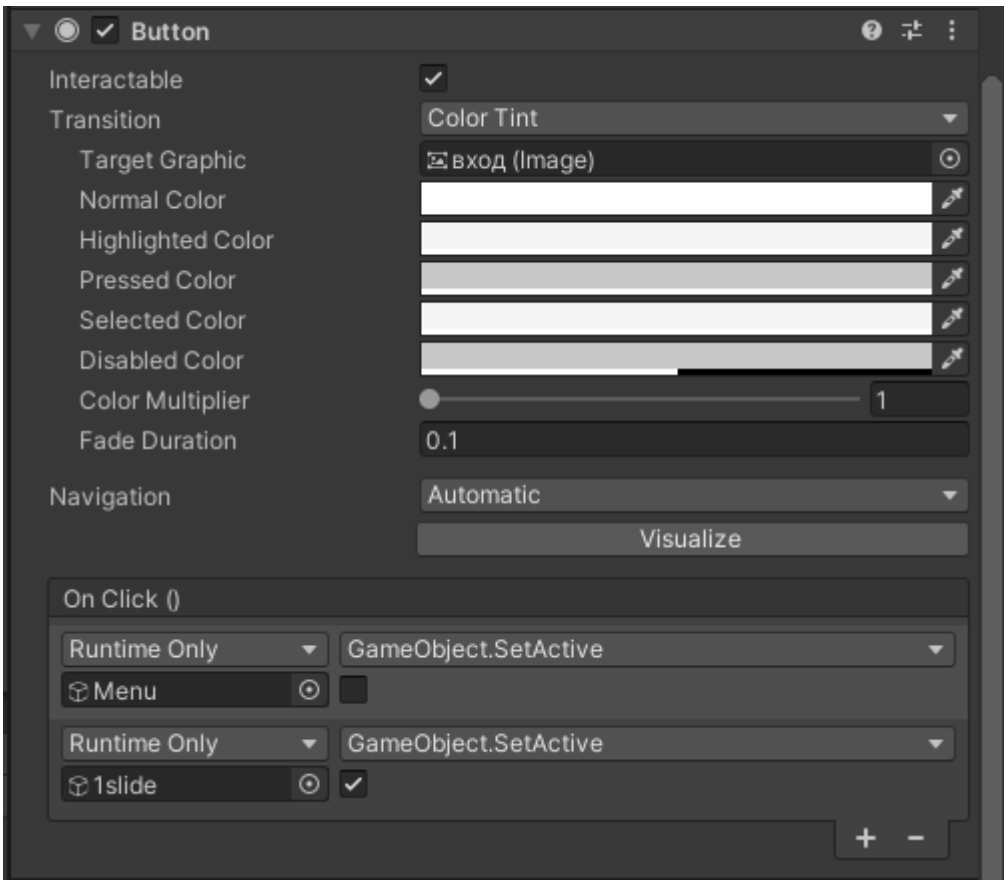


Рисунок 4.5 – Робота з логічними функціями в кнопках

Робота з тестами виконана таким чином, що при натисненні на неправильний варіант відповіді користувач отримує повідомлення про помилку та підказку з правильною відповіддю, при виборі правильного варіанту відповіді користувач отримує доступ до кнопки Далі для продовження роботи та відповідне повідомлення.

Ця функція також реалізована через логічні функції в кнопках, але замість перемикання сцен (слайдів) вмикається текстовий елемент з повідомленням та кнопка.

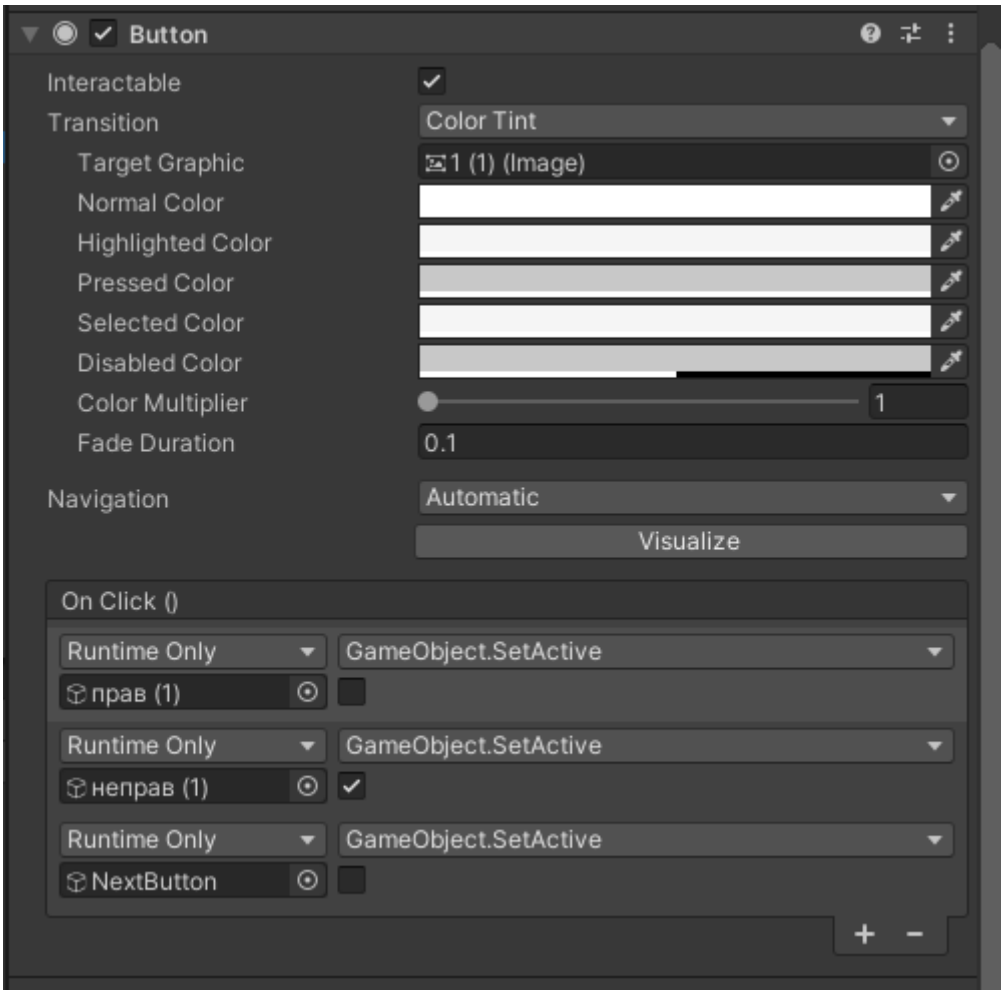


Рисунок 4.6 – Використання логічних функцій для кнопки з неправильним варіантом відповіді

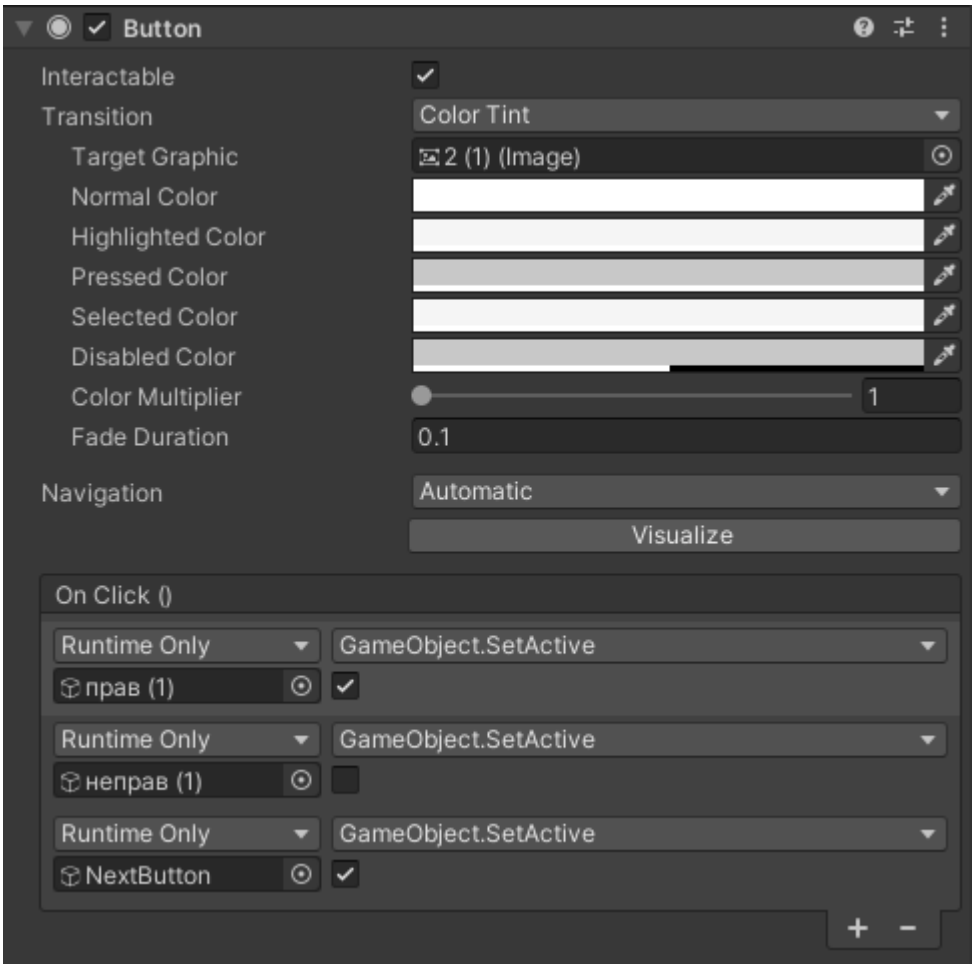


Рисунок 4.7 – Використання логічних функцій для кнопки з правильним варіантом відповіді

Кнопка повтору у кінці роботи з тренажером перенаправляє користувача до початкового екрану за допомогою логічних функцій, що дозволяє повторити роботу без перезавантаження тренажеру.

4.2 Інструкція по використанню програмного забезпечення

Після запуску застосунку користувач переходить до початкового екрану тренажеру. Найкраще розширення для комфортної роботи обирається автоматично згідно з налаштувань персонального комп’ютера.

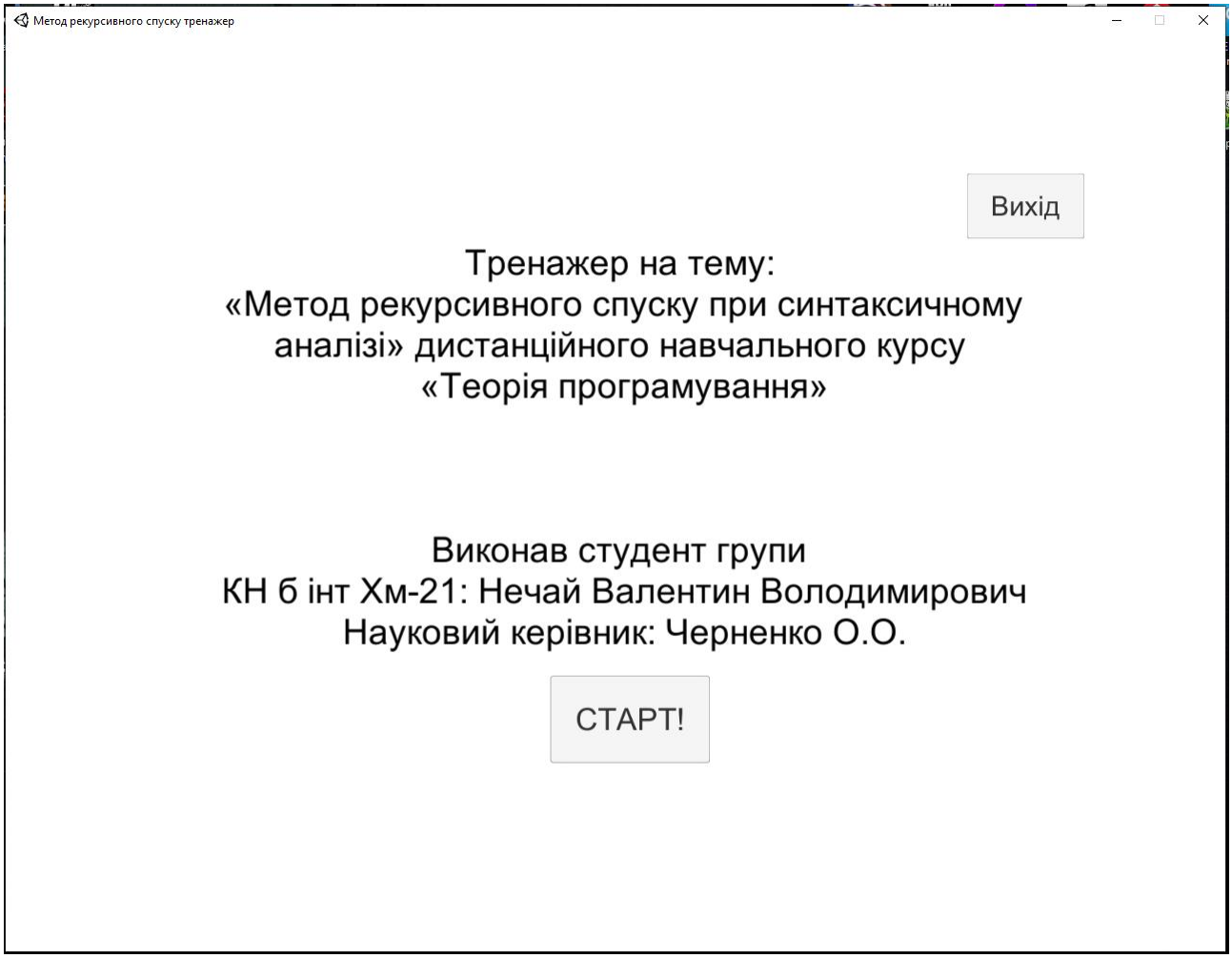


Рисунок 4.8 – Початковий екран тренажера

На даному слайді користувач бачить тему та виконавців роботи, може перейти до навчальних матеріалів через кнопку «СТАРТ» або закрити застосунок через кнопку «Вихід».

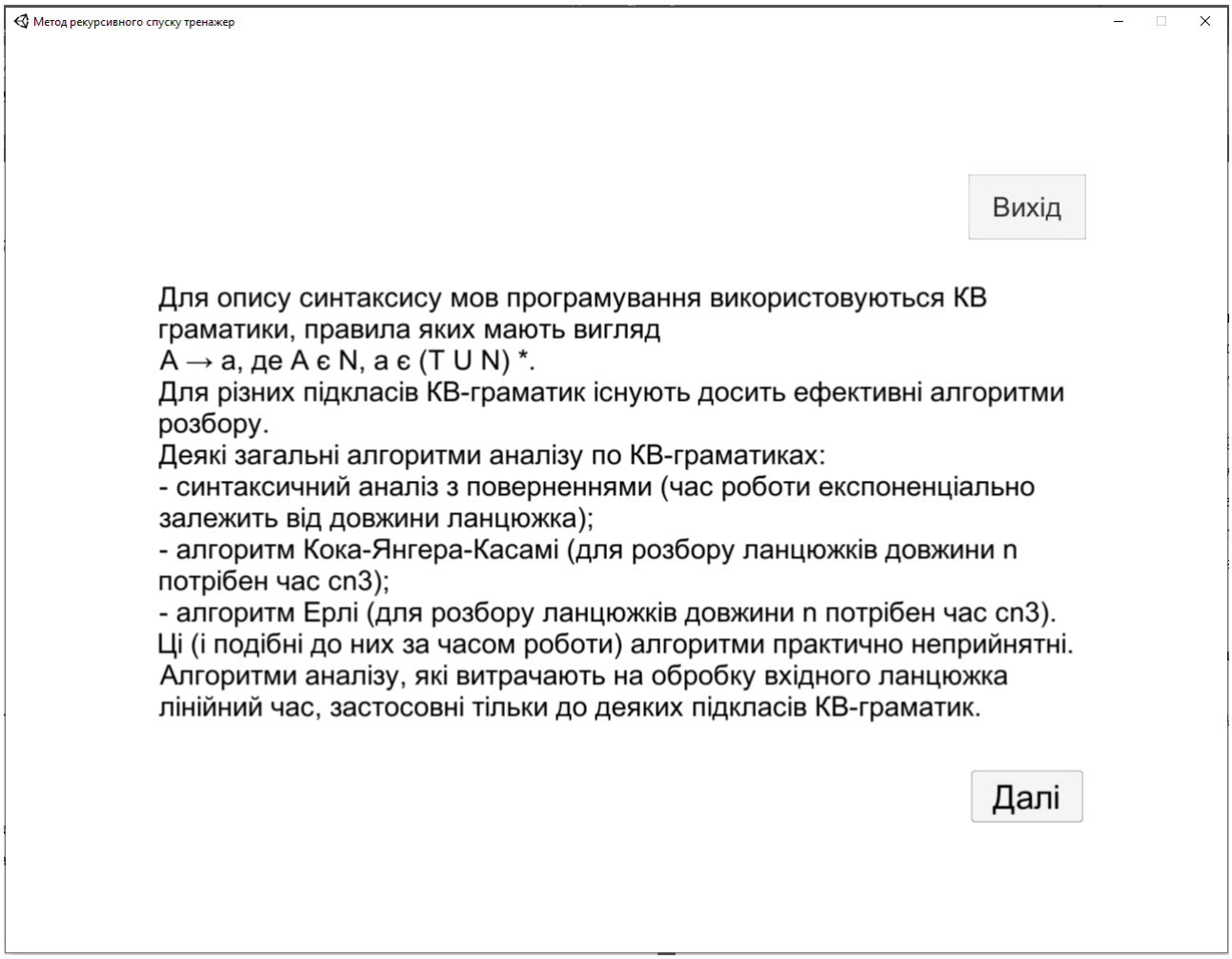


Рисунок 4.9 – Перший навчальний елемент тренажера з теоретичною інформацією

Після ознайомлення з теорією користувач переходить до тестових завдань з теми.

Під час роботи з тестовими завданнями користувач повинен обрати один з трьох варіантів відповіді (див. Рисунок 4.10).

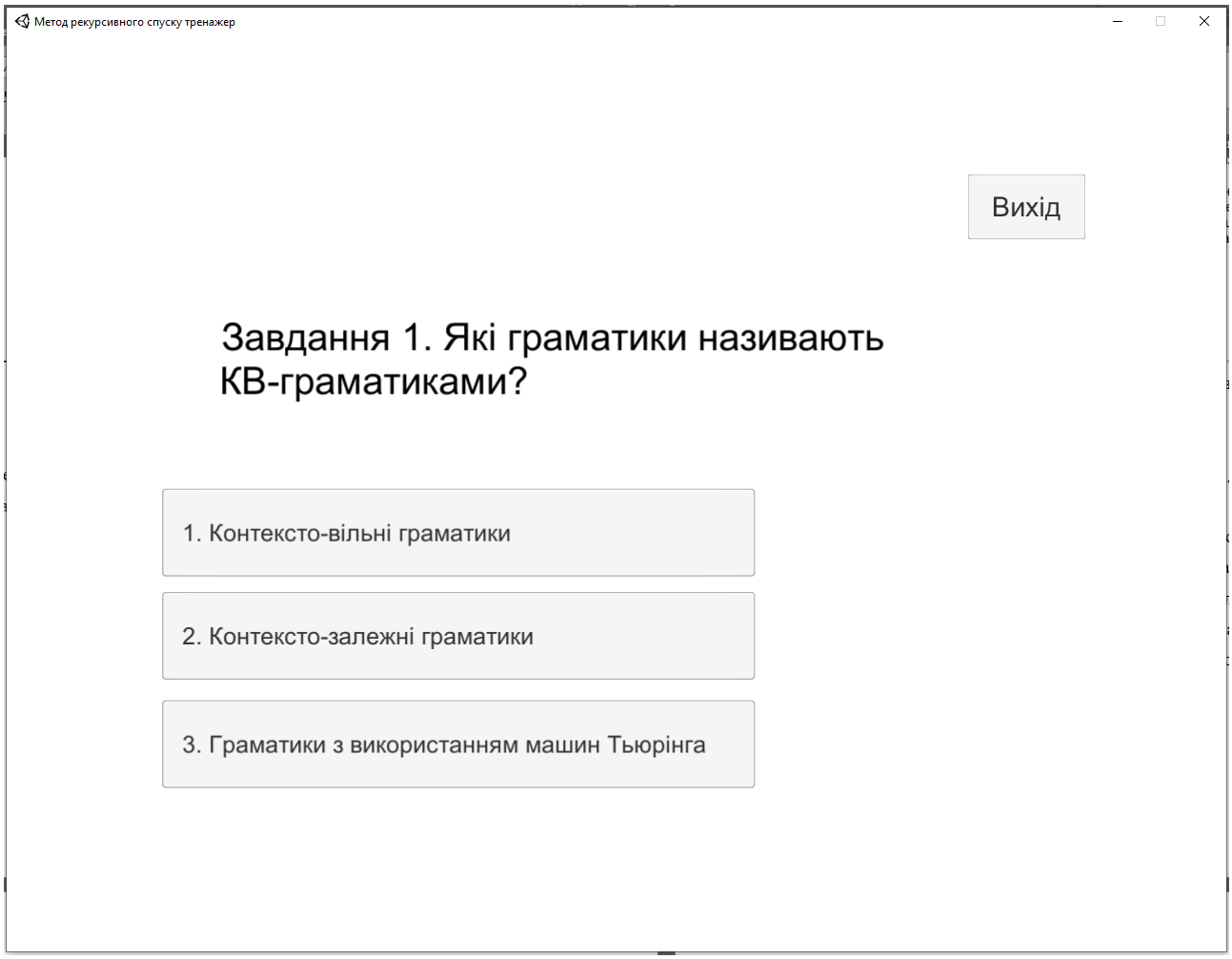


Рисунок 4.10 – Практичне завдання 1 з трьома варіантами відповіді

Після обрання неправильного варіанту відповіді користувач отримує відповідне повідомлення та підказку з правильним варіантом(див. Рисунок 4.11).

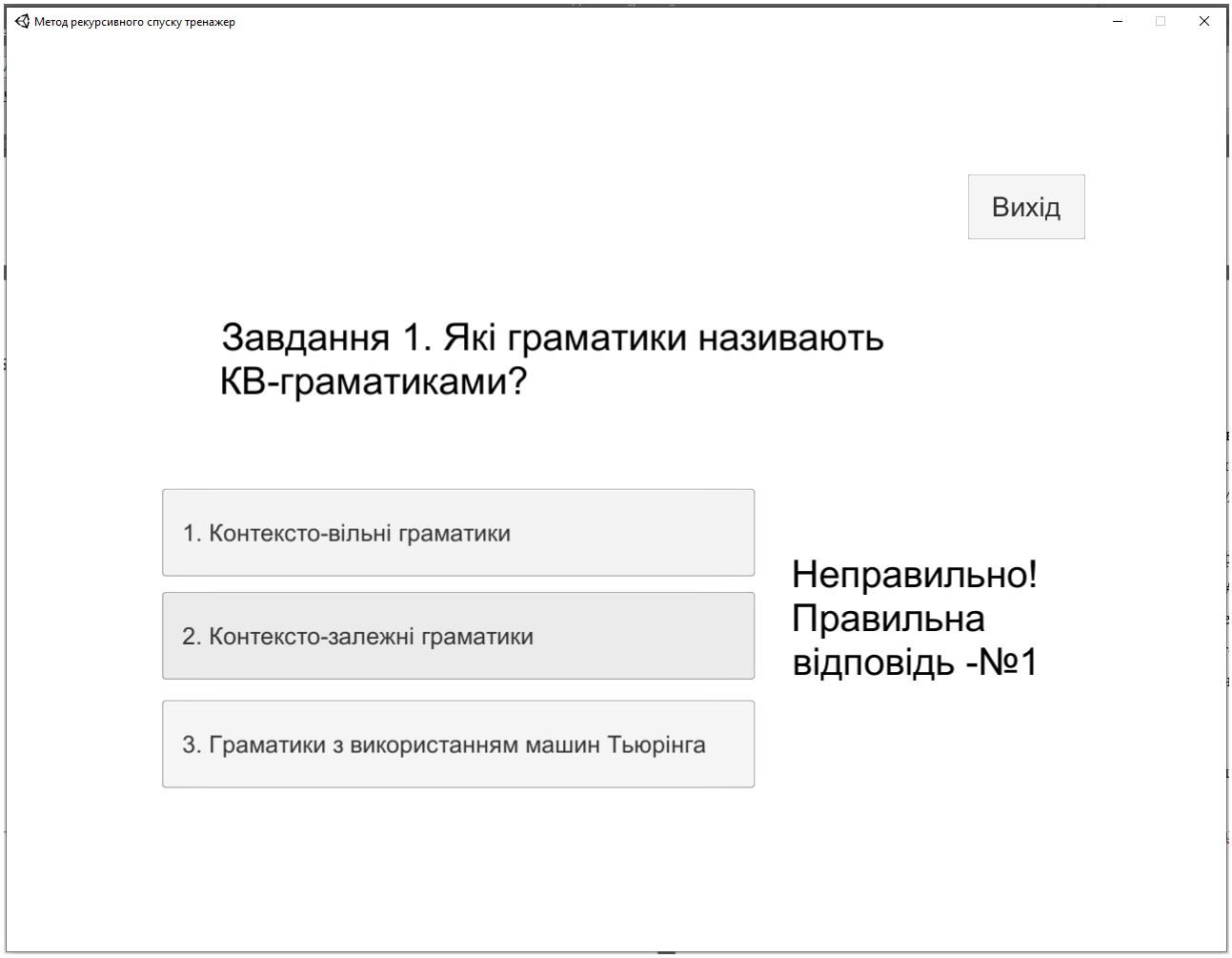


Рисунок 4.11 – Практичне завдання 1 після вибору неправильного варіанту відповіді

При виборі правильного варіанту користувач отримує доступ до кнопки «Далі» для продовження роботи та відповідне повідомлення(див. Рисунок 4.12).

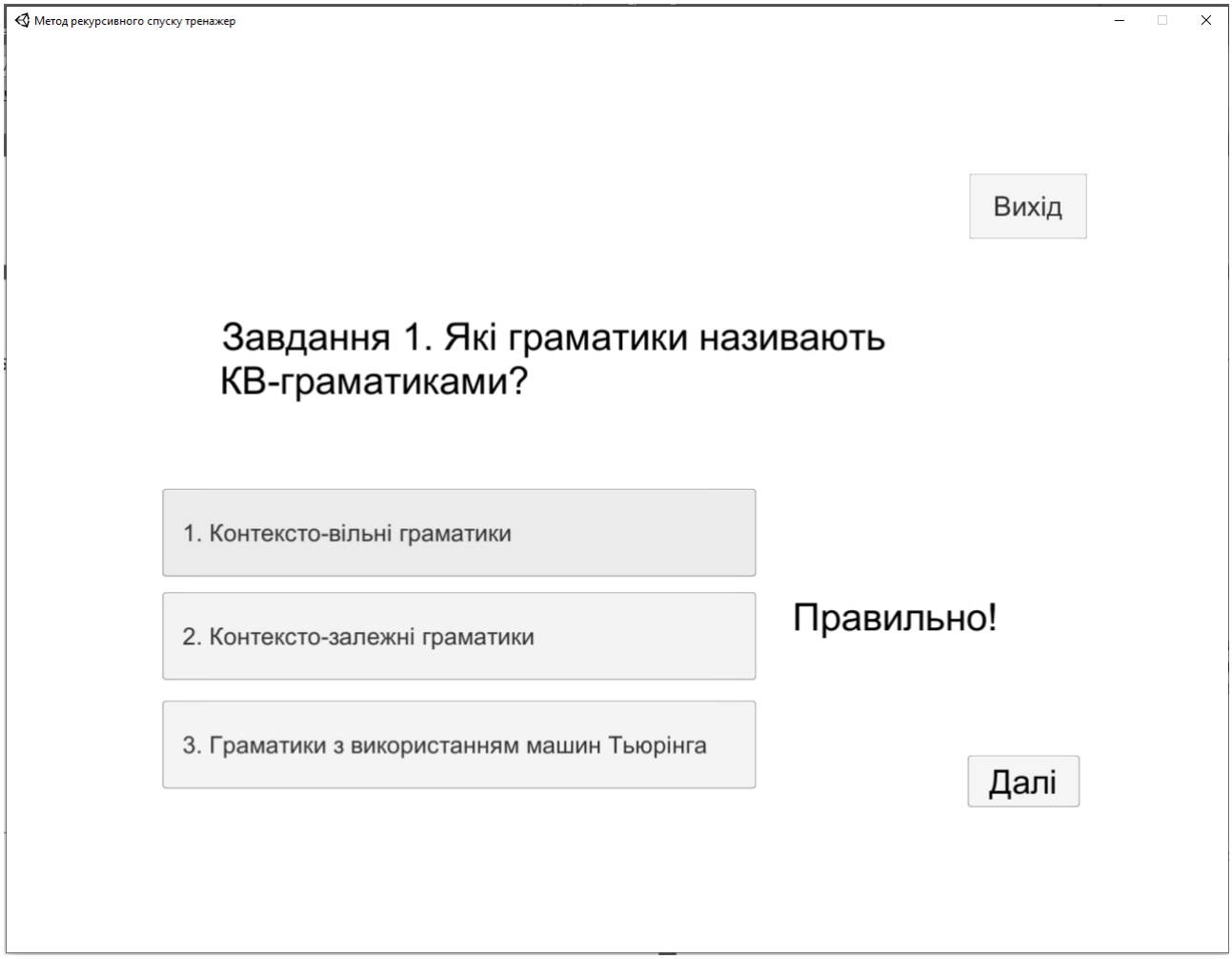


Рисунок 4.12 – Практичне завдання 1 після вибору правильного варіанту відповіді та поява кнопки «Далі»

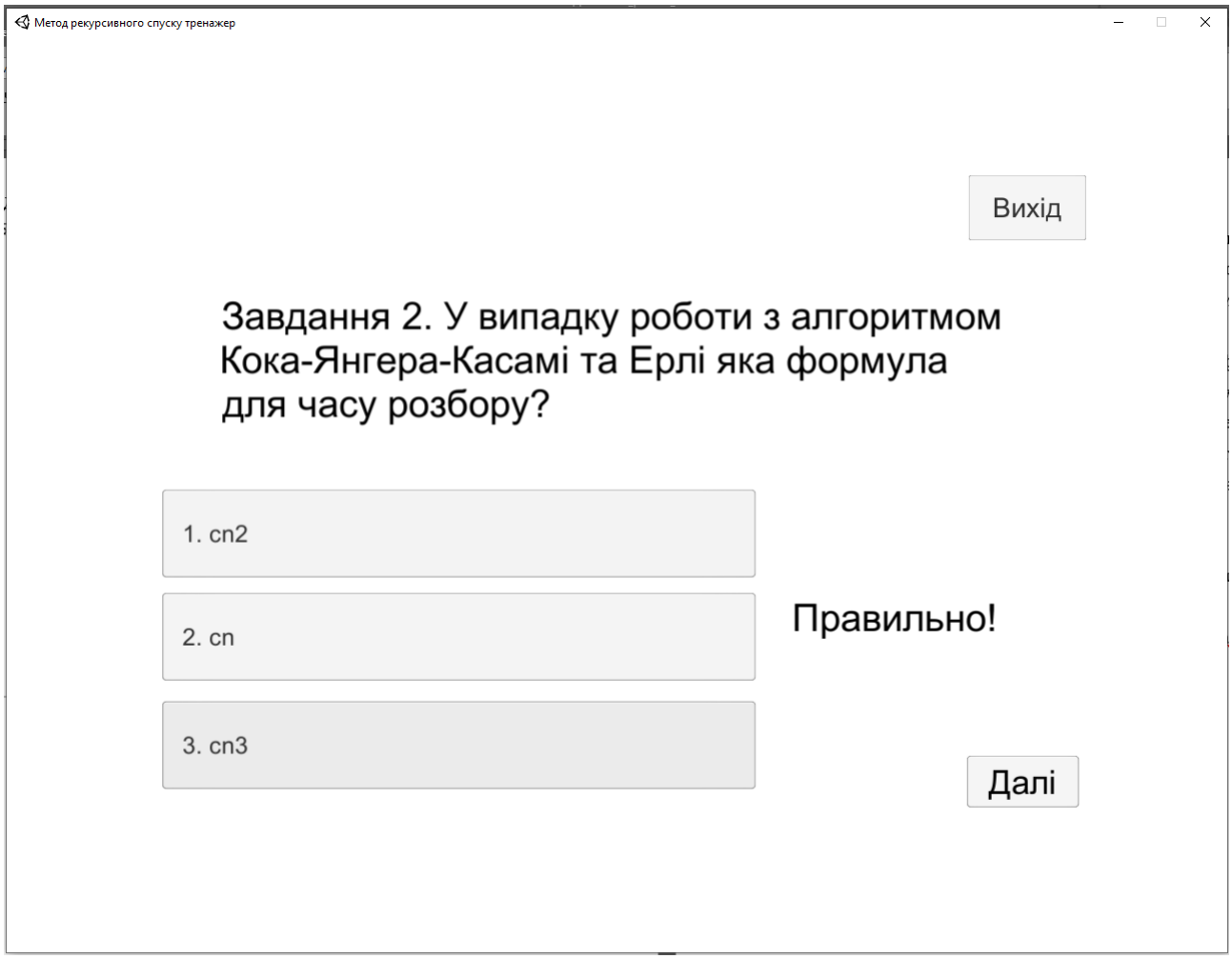


Рисунок 4.13 – Вибір правильної відповіді на Завдання 2

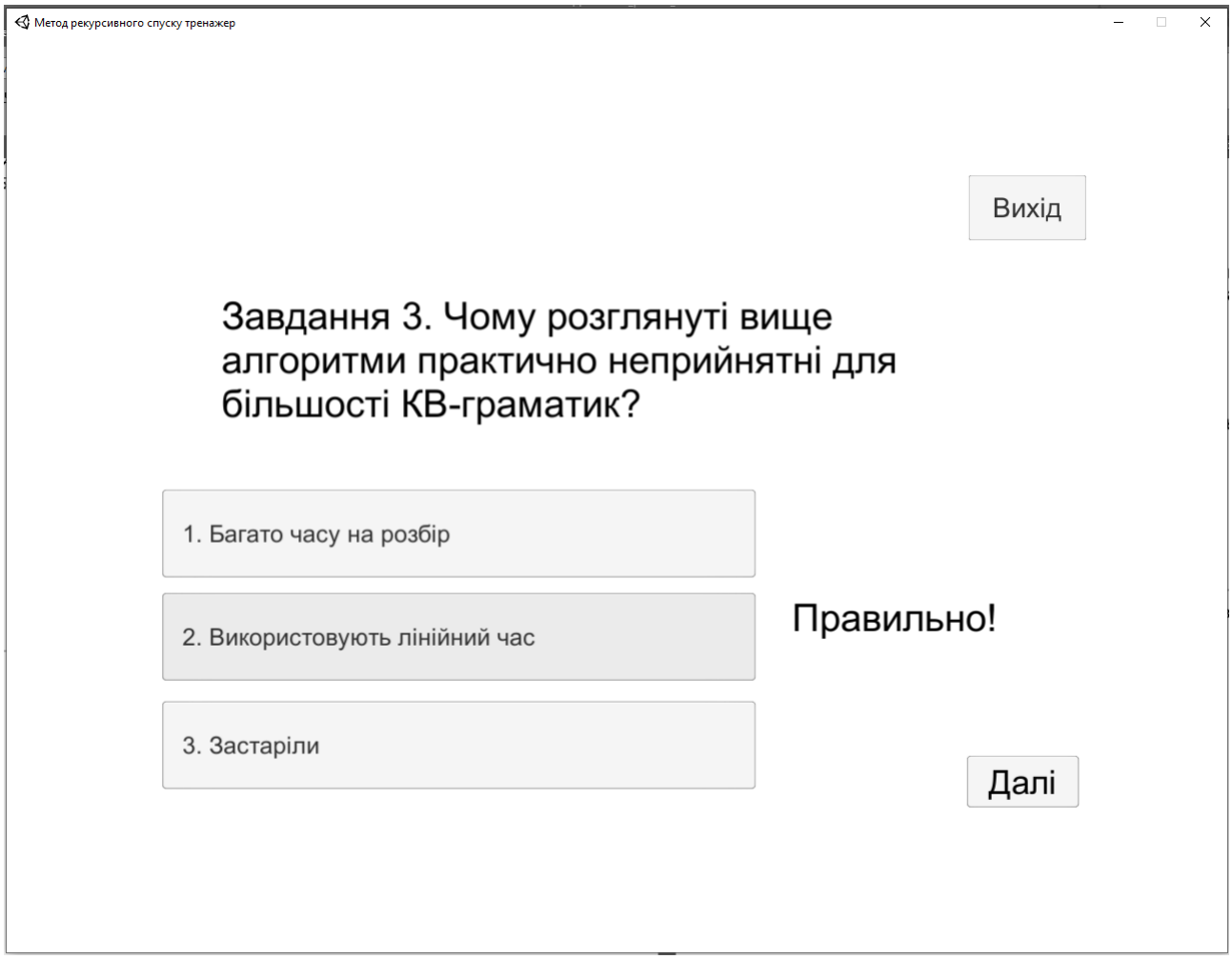


Рисунок 4.14 - Вибір правильної відповіді на Завдання 3

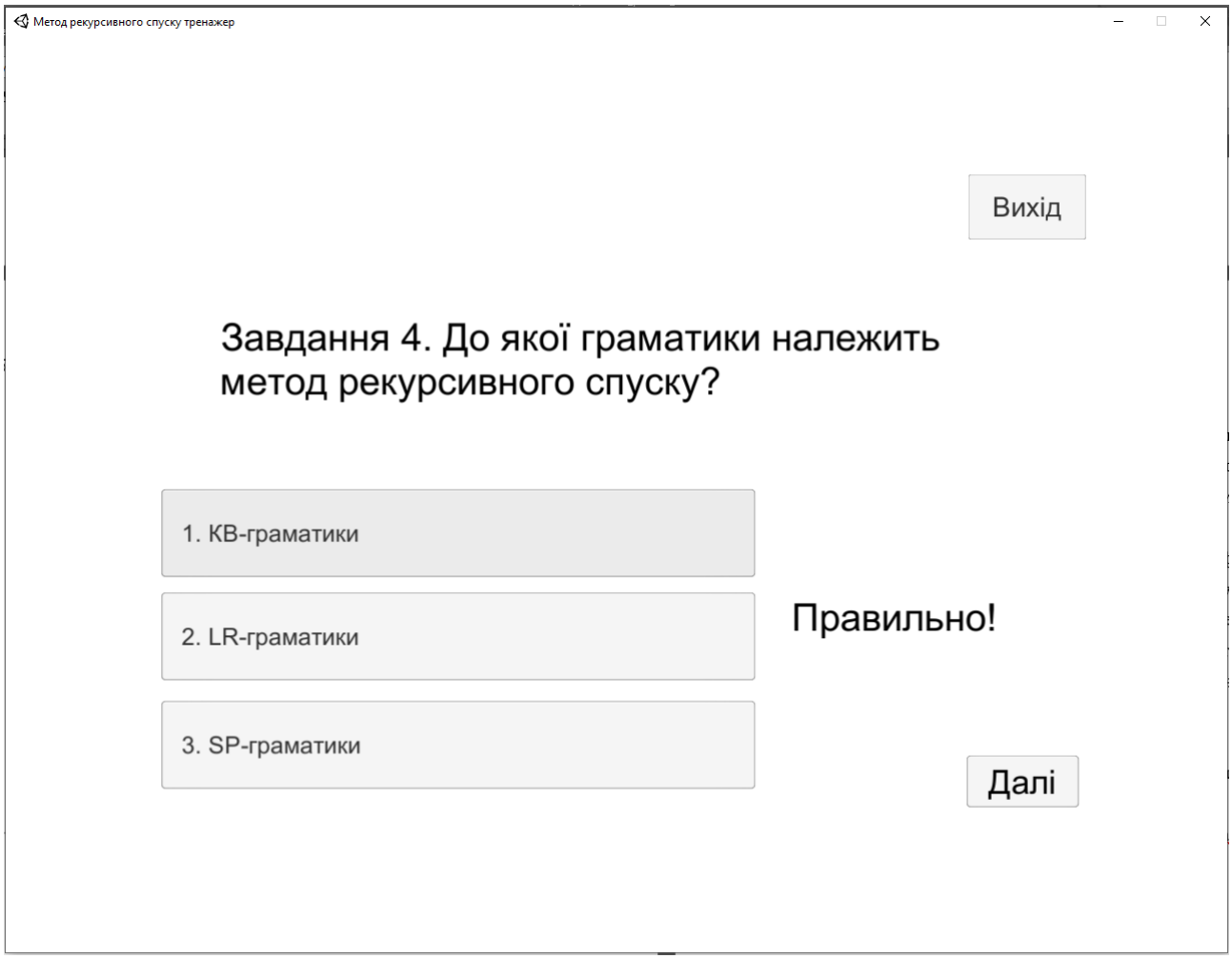


Рисунок 4.15 - Вибір правильної відповіді на Завдання 4

Після проходження практичних завдань за отриманим теоретичним матеріалом користувач отримує наступний теоретичний матеріал.

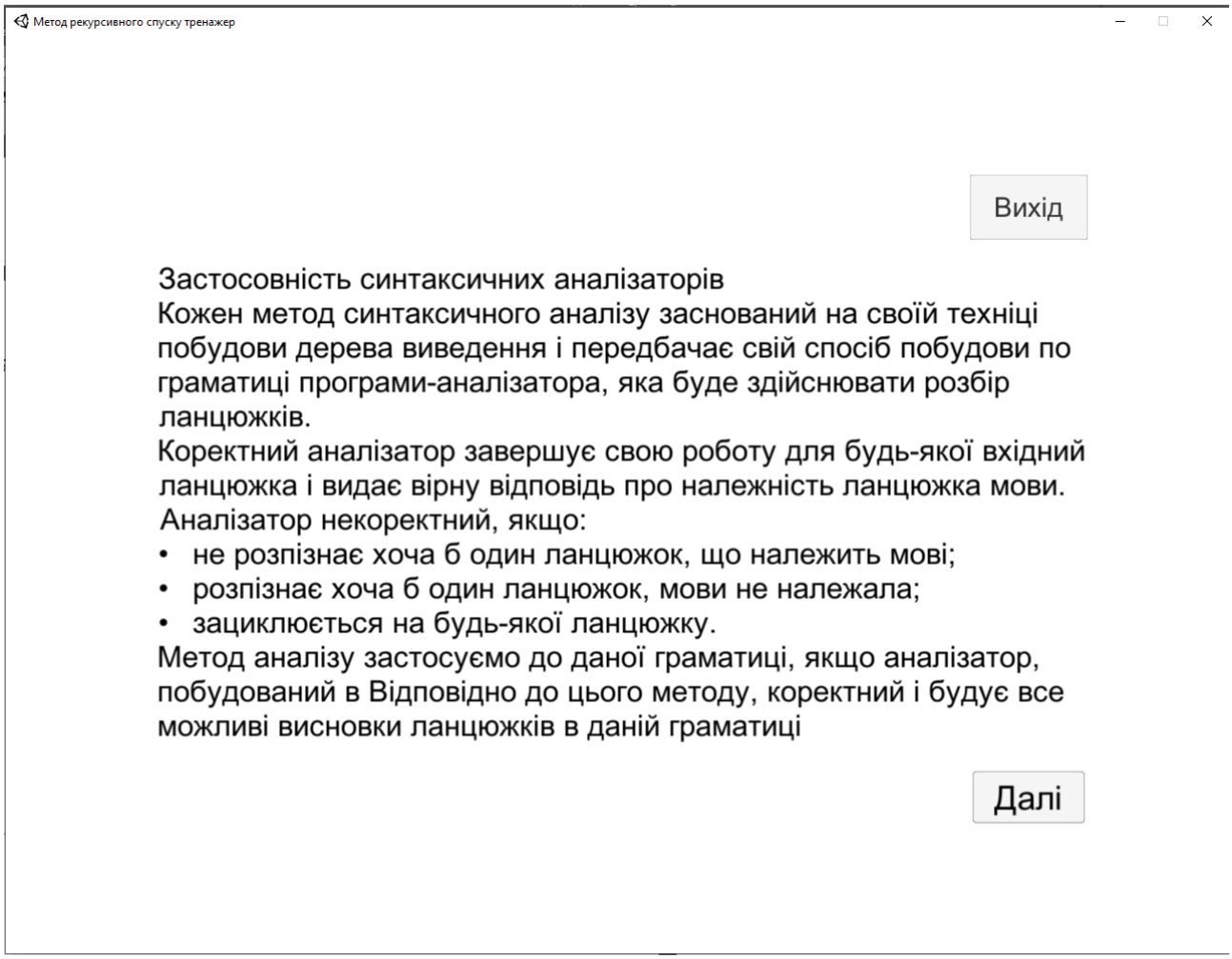


Рисунок 4.16 – Теоретичний матеріал до теми

Після ознайомлення з теоретичним матеріалом користувач переходить до тестів робота з якими типова до описаної вище.

Після завершення роботи з практичними завданнями користувач отримує теоретичний матеріал (див. Рисунок 4.17).

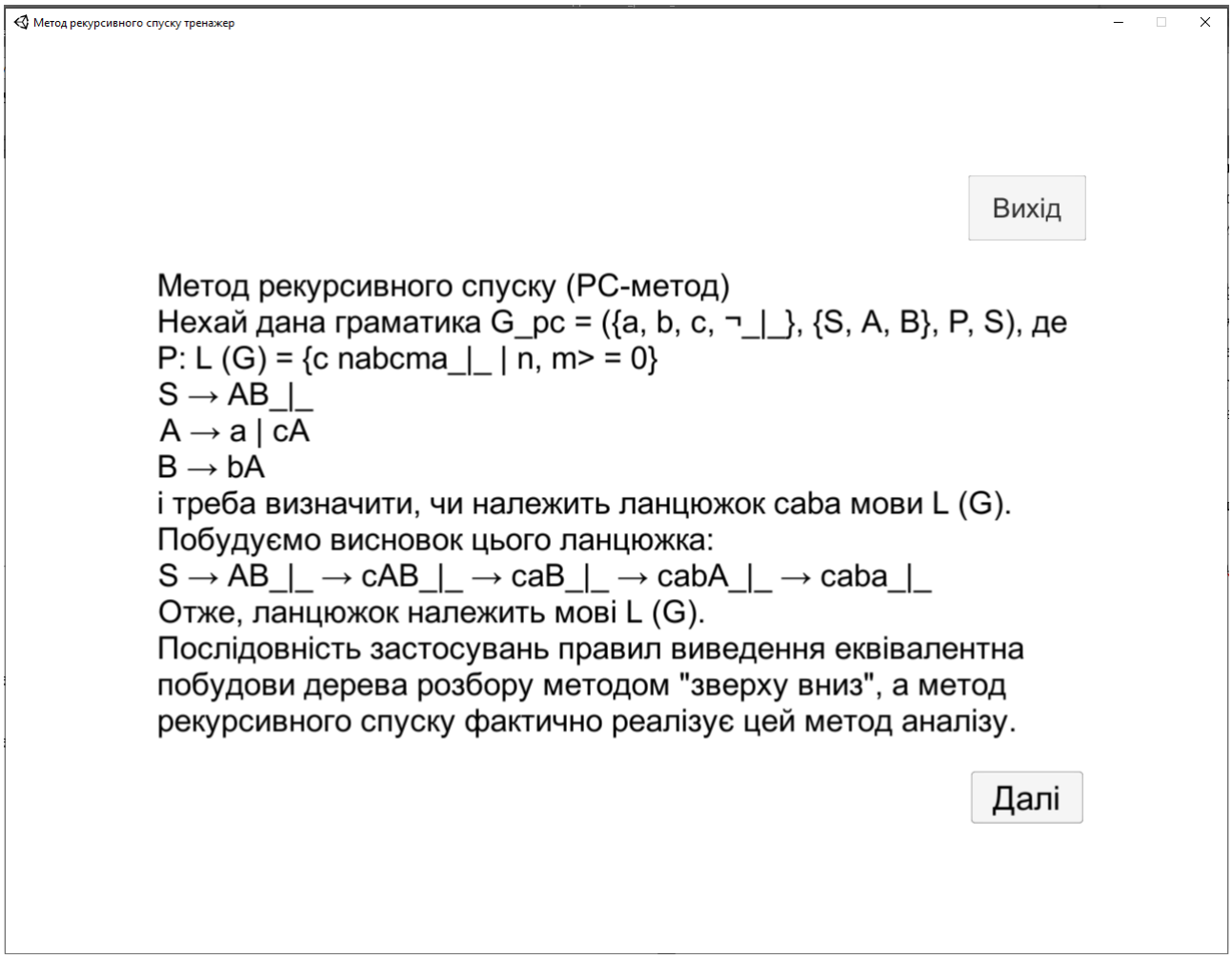


Рисунок 4.17 – Теоретичний матеріал до теми

Після ознайомлення з теоретичним матеріалом користувач переходить до практичних завдань в який варіантами відповіді виступають зображення.

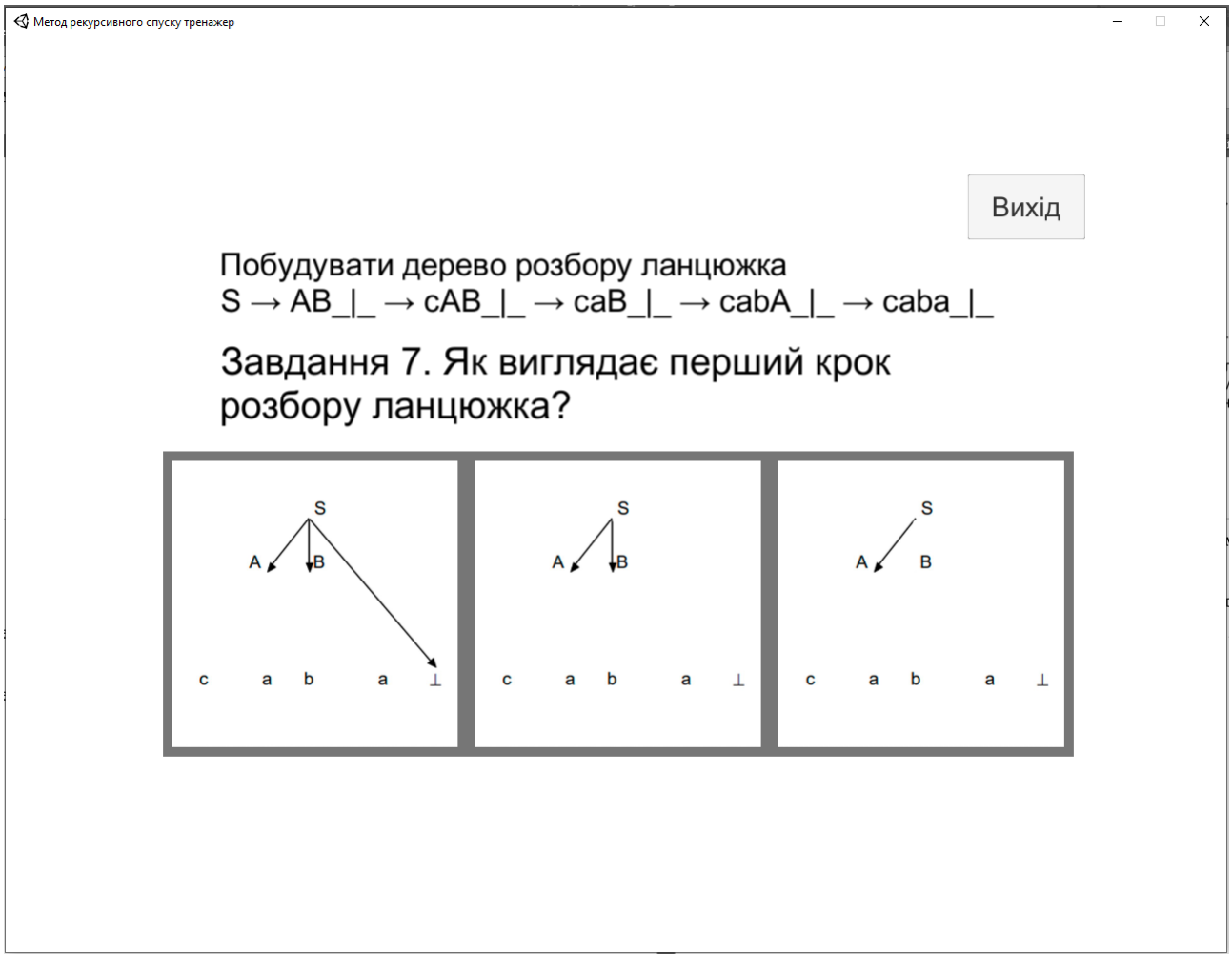


Рисунок 4.18 – Практичне завдання з варіантами відповіді у вигляді зображення

Різниця між практичне завдання з варіантами відповіді у вигляді зображення і звичайним тестом заключається лише у типі відображення. Після вибору неправильної відповіді користувач отримує повідомлення та підказку (див. Рисунок 4.19).

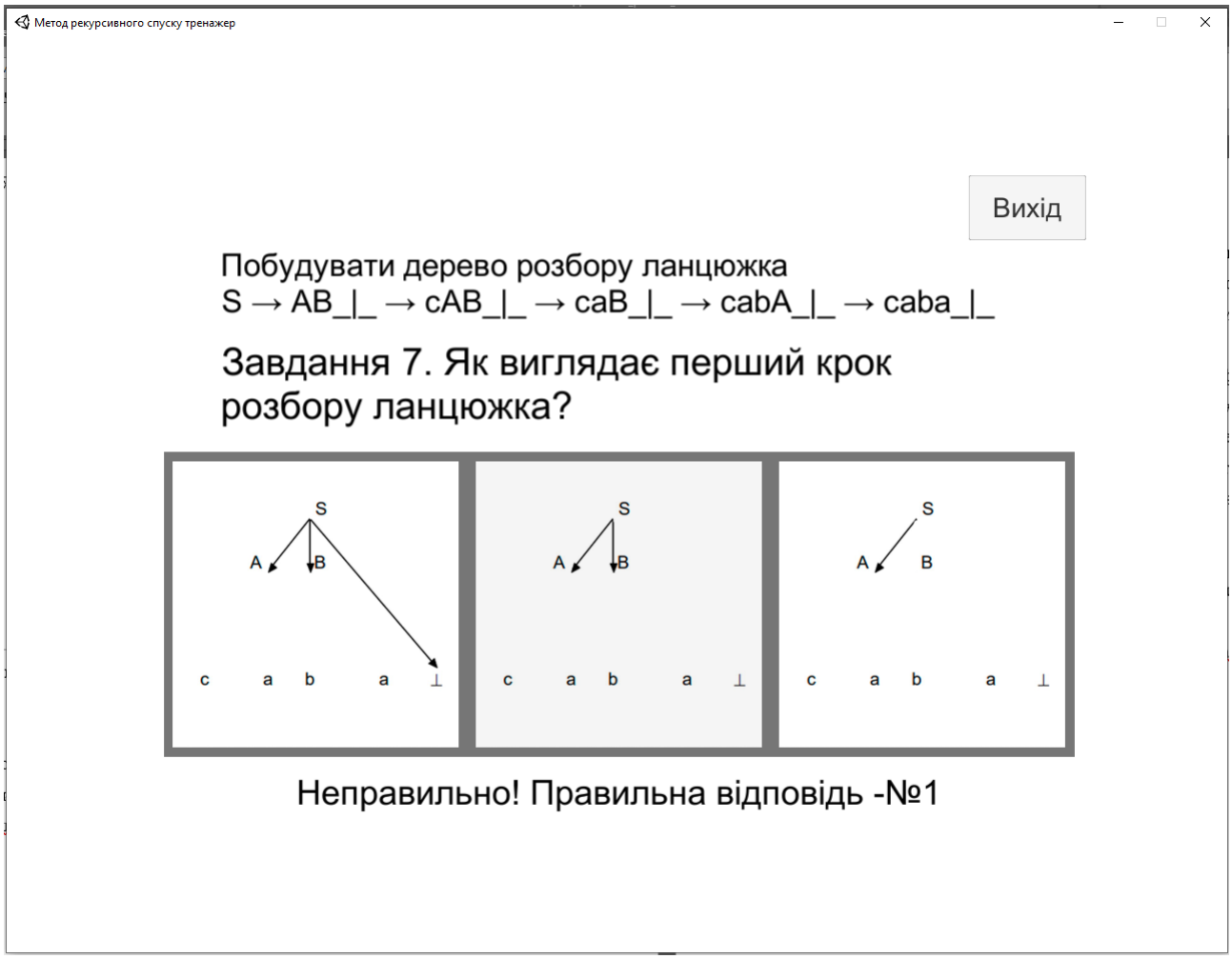


Рисунок 4.19 - Практичне завдання з варіантами відповіді у вигляді зображення після вибору неправильної відповіді

Після вибору правильної відповіді користувач отримує повідомлення та доступ до кнопки «Далі» (див. Рисунок 4.20).

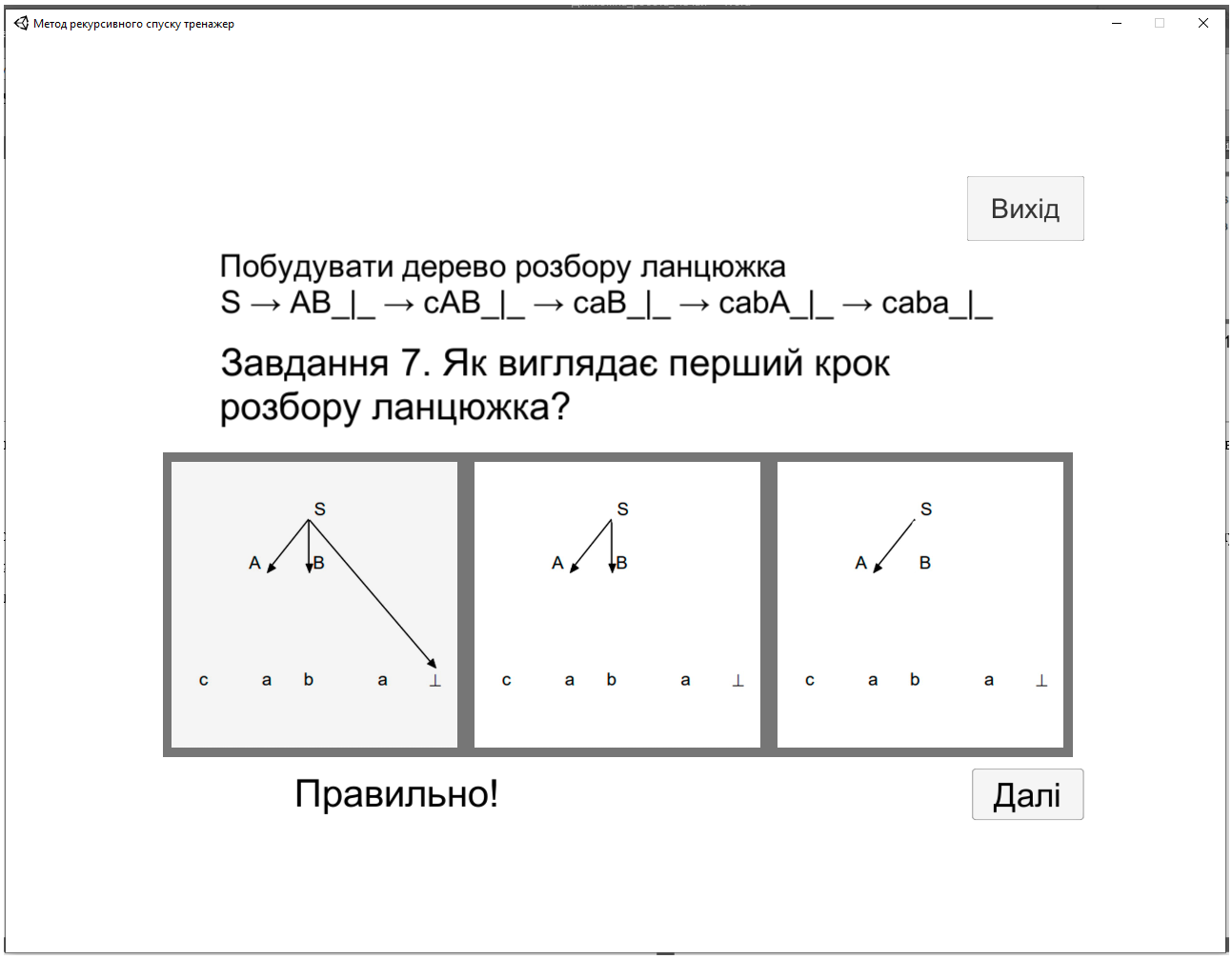


Рисунок 4.20 - Практичне завдання з варіантами відповіді у вигляді зображення після вибору правильної відповіді

Робота з наступними кроками розбору даного прикладу типова. Після закінчення роботи з навчальними тренажерами користувач переходить до вікна з кнопкою повтору та з інформацією про успішне завершення тренажеру.

Рисунок 4.21 – Завершення роботи з тренажером та кнопка «Повторити роботу?»

ВИСНОВКИ

Під час роботи над дипломною роботою було створено алгоритм роботи тренажеру та блок-схему для цього алгоритму, тренажер для навчання та інструкцію з його використання. В алгоритмі наведено довідковий матеріал та практичні завдання у вигляді тестів з теми «Метод рекурсивного спуску при синтаксичному аналізі».

Програмно реалізовано програму-тренажер на платформі Unity 2020 за допомогою MS Visual Studio 2019 та мови програмування C#.

Головними позитивними сторонами розробленого тренажеру є:

1. Наявність як практичного так і теоретичного матеріалу;
2. Видача теоретичного матеріалу у порядку його необхідності під час роботи з практичними матеріалом;
3. Перевірка введеної відповіді та видача підказки;
4. Можливість повторити роботу з тренажером через відповідну кнопку.

По результатам роботи з бакалаврською роботою було знайдено теоретичну інформацію для звіту, знайдено та оформлено завдання в тренажер, розроблено алгоритм та блок-схеми алгоритму роботи з тренажером, програмно реалізовано тренажер, дотримано вимоги оформлення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ємець О.О. Методичні рекомендації до виконання бакалаврської роботи для студентів спеціальності 122 «Комп'ютерні науки та інформаційні технології» освітня програма «Комп'ютерні науки» галузь знань – 12 «Інформаційні технології»/ О.О. Ємець,–Полтава; ПУЕТ, 2017, – 71 с.

2. Алексов, С.В. Пояснювальна записка до бакалаврської роботи на тему Алгоритмізація та програмування тренажера «Дерева розбору» дистанційного навчального курсу «Теорія програмування» /Алексов, С.В. [Електронний ресурс].– Режим доступу до ресурсу: http://dspace.puet.edu.ua/handle/123456789/8998

3. Скромінський, М.В. Пояснювальна записка до бакалаврської роботи на тему Розробка програмного забезпечення для тренажеру з теми «Контекстовільні граматики» дистанційоного навчального курсу «Теорія програмування» /Скромінський, М.В. [Електронний ресурс].– Режим доступу до ресурсу: http://dspace.puet.edu.ua/handle/123456789/9025

4. Черненко О.О. Теорія програмування, Синтаксичний аналіз [Електронний ресурс].– Режим доступу до ресурсу: http://www2.el.puet.edu.ua/st/mod/page/view.php?id=105325

5. Рекурсивний спуск [Електронний ресурс].– Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D1%81%D0%B8%D0%B2%D0%BD%D0%B8%D0%B9_%D1%81%D0%BF%D1%83%D1%81%D0%BA

6. Unity (рушій гри), [Електронний ресурс].– Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Unity_(%D1%80%D1%83%D1%88%D1%96%D0%B9_%D0%B3%D1%80%D0%B8)

7. Шворoб І. Б, Порівняльний аналіз методів синтаксичного розбору, [Електронний ресурс].– Режим доступу до ресурсу:

http://science.lpnu.ua/sites/default/files/journalpaper/2017/jun/2654/814ism2015min-197-202.pdf

8. Рекурсивный спуск [Електронний ресурс].– Режим доступу до ресурсу: https://dic.academic.ru/dic.nsf/ruwiki/1119916

9. Метод рекурсивного спуска (РС-метод) [Електронний ресурс].– Режим доступу до ресурсу: cmcmsu.info/download/she.programming.languages.L12-13.slides.pdf+&cd=5&hl=ru&ct=clnk&gl=ua

10. Unity, Робота з платформою [Електронний ресурс] .– Режим доступу до ресурсу: https://unity.com/ru

11. Unity [Електронний ресурс].– Режим доступу до ресурсу: https://learn.unity.com/

12. Вивчення мови програмування C# [Електронний ресурс].– Режим доступу до ресурсу:

https://docs.microsoft.com/en-us/visualstudio/get-started/csharp/?view=vs-2019

13. Unity, Робота з платформою [Електронний ресурс].– Режим доступу до ресурсу: http://39.104.106.62/Manual/OfflineDocumentation.html

14. Метод рекурсивного спуску [Електронний ресурс].– Режим доступу до ресурсу: https://ppt-online.org/181333

ДОДАТОК А

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;
using UnityEngine;

public class QuitButton : MonoBehaviour
{
   public void QuitApp()
   {
      Application.Quit();
   }
void Start()
   {
      Button btn = nxtButton.GetComponent<Button>();
      btn.onClick.AddListener(TaskOnClick);
   }

void TaskOnClick()
   {
      Theme1.SetActive(false);
      Theme2.SetActive(true);
   }

void Start()
   {
      Button btn = menuButton.GetComponent<Button>();
      btn.onClick.AddListener(TaskOnClick);
   }
```

```
void TaskOnClick()
    {
        Slide8.SetActive(false);
        MainMenu.SetActive(true);
    }
public void Next()
    {
        Input.SetActive(false);
        Input2.SetActive(true);
        ShowInpTxt.text = inputedTxt.text;
    }
}
```