**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)

«_____»_____2020 р.

# ПОЯСНЮВАЛЬНА ЗАПИСКА ДО ДИПЛОМНОЇ РОБОТИ

**на тему**

# РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ТРЕНАЖЕРА ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ТЕОРІЯ ІНФОРМАЦІЇ ТА КОДУВАННЯ» З ТЕМИ «КОДИ ІЗ ВИЯВЛЕННЯМ ПОМИЛОК»

**зі спеціальності 122 «Комп'ютерні науки»**

**Виконавець роботи** Жайворонок Ярослав Ігорович_____«___»_____2020 р.
(підпис)

**Науковий керівник** , к. ф.-м. н., доц., Парфьонова Тетяна Олександрівна

__ _____«___»_____2020 р.
(підпис)

**ПОЛТАВА 2020 р.**

**ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ**
**«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»**

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри _____ О.О. Ємець**
(підпис)

«1» вересня 2020 р.

**ЗАВДАННЯ ТА КАЛЕНДАРНИЙ ГРАФІК**
**ВИКОНАННЯ ДИПЛОМНОЇ РОБОТИ**

**Студент спеціальності 122 «Комп'ютерні науки»**
**Прізвище, ім'я, по батькові Жайворонок Ярослав Ігорович**
1. **Тема «Розробка програмного забезпечення для тренажера дистанційного навчального курсу «Теорія інформації та кодування» з теми «Коди із виявленням помилок»**
**затверджена наказом ректора № 115-Н від «01» вересня 2020 р.**
Термін подання студентом дипломної роботи «___»_____ 2020 р.
2.Вихідні дані до дипломної роботи курс лекцій з «Теорії інформації та кодування» та інформаційні джерела

3. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ
ВСТУП
1. ПОСТАНОВКА ЗАДАЧІ
2. ІНФОРМАЦІЙНИЙ ОГЛЯД
2.1. Огляд інших схожих тренажерів
2.2. Переваги оглянутих робіт
2.3. Недоліки оглянутих робіт
2.4. Підсумки проведеного інформаційного огляду
3. ТЕОРЕТИЧНА ЧАСТИНА
3.1. Обґрунтування актуальності розробки тренажеру
3.2. Алгоритм тренажер
3.3. Блок-схема алгоритму тренажера
4. ПРАКТИЧНА ЧАСТИНА
4.1. Опис коду тренажеру
4.2. Опис роботи тренажеру
4.3. Інструкція для запуску тренажеру на різних пристроях
4.4. Перевірка працездатності тренажеру
ВИСНОВКИ

4. Перелік графічного матеріалу (з точним визначенням кількості блок-схем, іншого графічного матеріалу) 5-6 аркушів блок-схем, та необхідні ілюстрації

5. Консультанти розділів дипломної роботи

| Розділ | Прізвище, ініціали, посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Вступ | Парфьонова Т.О., доц., к.ф.-м.н. | | |
| Постановка задачі | Парфьонова Т.О., доц., к.ф.-м.н. | | |
| Інформаційний огляд | Парфьонова Т.О., доц., к.ф.-м.н. | | |
| Теоретична частина | Парфьонова Т.О., доц., к.ф.-м.н. | | |
| Практична частина | Парфьонова Т.О., доц., к.ф.-м.н. | | |

6. Календарний графік виконання дипломної роботи

| Зміст роботи | Термін виконання | Фактичне виконання |
|---|---|---|
| Вступ | | |
| Вивчення методичних рекомендацій та стандартів та звіт керівнику | | |
| Постановка задачі | | |
| Інформаційний огляд джерел бібліотек та інтернету | | |
| Теоретична частина | | |
| Практична частина | | |
| Закінчення оформлення | | |
| Доповідь студента на кафедрі | | |
| Доробка (за необхідністю), рецензування | | |

Дата видачі завдання «1» вересня 2020 р.

Студент_____
(підпис)

Науковий керівник _____ доц., к.ф.-м.н., Т.О. Парфьонова
(підпис) (науковий ступінь, вчене звання, ініціали та прізвище)

***Результати захисту дипломної роботи***

Дипломна робота оцінена на _____
(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК № ____ від «____» _____ 2020 р.

Секретар ЕК_____ _____
(підпис) (ініціали та прізвище)

# РЕФЕРАТ

**Записка:** 113 с., 40 рис., 2 додатки (на 46 сторінках), 11 джерел.

**Предметом розробки** є тренажер для дистанційного курсу «Теорія інформації та кодування» з теми «Коди із виявленням помилок».

**Мета роботи** – побудувати алгоритм та запрограмувати навчальний тренажер з теми «Коди із виявленням помилок» дистанційного навчального курсу «Теорія інформації та кодування».

**Методи розробки.** Для створення навчального тренажеру використано багатоплатформовий інструмент Unity та об'єктно–орієнтована мова програмування – C# [1]. Для створення алгоритму тренажеру використано двійкові та недвійкові коди, що виявляють помилки .

**Розроблено** алгоритм тренажеру та сам тренажер для навчання студентів напряму «Комп’ютерні науки» теми «Коди із виявленням помилок».

Програмно реалізовано алгоритм тренажеру.

В результаті проведеного аналізу працездатності тренажеру виявлено, що він працює коректно.

Тренажер впроваджено у дистанційний курс ПУЕТ.

Ключові слова: КОДИ ІЗ ВИЯВЛЕННЯМ ПОМИЛОК, ТРЕНАЖЕР, ДВІЙКОВІ КОДИ, НЕДВІЙКОВІ КОДИ

**ЗМІСТ**

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

| Умовні позначення, символи, скорочення, терміни | Пояснення умовних позначень, символів, скорочень |
|---|---|
| $d_{min}$ | Мінімальна кодова відстань |
| $k$ | Кількість інформаційних елементів у комбінації. |
| $m$ | Число одиниць у комбінації. |
| $n$ | Повна довжина комбінації. |
| $r$ | Кількість перевірних елементів у закодованій комбінації. |
| $R_{\text{над}}$ | Надмірність побудованої комбінації |
| $w$ | Кількість одиниць у кодовій комбінації (вага). |

# ВСТУП

Актуальність цієї роботи полягає в розробці застосунку (навчального тренажеру) в дистанційний курс ПУЕТ для студентів фаху «Комп'ютерні науки», які вивчають в дисципліні «Теорія інформації та кодування» коди із виявленням помилок.

За допомогою тренажеру студенти зможуть вивчати коди із виявленням помилок не виходячи з домівки. Розроблена гнучка структура надасть змогу студентам отримувати всі методичні рекомендації та підказки для повного розуміння матеріалу. Також даний тренажер не було знайдено у відомих дистанційних курсах, тому його створення є важливим для студентів, як денної, так і для заочної (дистанційної) форми навчання.

Мета роботи – побудувати алгоритм та запрограмувати навчальний тренажер з теми «Коди із виявленням помилок» дистанційного навчального курсу «Теорія інформації та кодування».

Для досягнення мети були поставлені такі завдання:

- ознайомлення з методичними вказівками та стандартами оформлення магістерських робіт;
- здійснити постановку задачі;
- детальний огляд літератури, що використовується для вивчення кодів із виявленням помилок;
- структуризація вивченого матеріалу;
- розробка алгоритму навчального тренажеру;
- розробка блок–схеми алгоритму тренажеру;
- вибір найкращої платформи для програмної реалізації тренажеру;
- розробка тренажеру на обраній платформі;
- перевірка працездатності та правильності роботи тренажеру;
- оформлення висновків.

Об'єктом розробки є програмне забезпечення для дистанційного навчання з курсу «Теорії інформації та кодування».

Предметом розробки є тренажер для дистанційного курсу «Теорія інформації та кодування» з теми «Коди із виявленням помилок».

Методи розробки. Для створення навчального тренажеру використано багатоплатформовий інструмент Unity та об'єктно–орієнтована мова програмування – C# [1]. Для створення алгоритму тренажеру використано двійкові та недвійкові коди, що виявляють помилки [2].

Новизною магістерської роботи є навчальний тренажер з теми «Розробка програмного забезпечення для тренажера дистанційного навчального курсу «Теорія інформації та кодування» з теми «Коди із виявленням помилок». Розроблений тренажер знаходиться не тільки у дистанційному курсі ПУЕТ, а і в інтернеті. Для полегшення доступу до тренажеру реалізована підтримка трьох платформ: Android (ver. 5.1 або вище), Windows, HTML5 (web), тому кожен бажаючий може завантажити та встановити цей тренажер скориставшись інструкцією в пункті(4.3).

Практичною цінністю є те, що кожен студент зможе вивчати коди із виявленням помилок без викладача, отримуючи при цьому, як теоретичні, так і практичні навички. Даний тренажер впроваджено в дистанційний курс з «Теорії інформації та кодування» в Полтавському університеті економіки та торгівлі.

Пояснювальна записка до магістерської роботи містить в собі: вступ; постановку задачі; інформаційний огляд, в якому розглянуто роботи інших студентів; теоретичну частину, де наведено побудований алгоритм для тренажера та його блок–схему; практичну частину, де описано код програми, її роботу та сформовано інструкцію з використання; висновки; літературу; два додатки.

Обсяг пояснювальної записки 63 сторінки, в які входить постановка задачі, теоретична частина, практична частина, 40 рисунків, список літератури – 11 джерел.

# 1 ПОСТАНОВКА ЗАДАЧІ

## 1.1. Постановка задачі

Задачею магістерської роботи є алгоритмізація та розробка програми — тренажеру з теми «Коди із виявленням помилок» для полегшення вивчення та поглиблення, закріплення знань з предмету «Теорія інформації та кодування».

Алгоритм, за яким студенти будуть вивчати коди із виявленням помилок повинен містити в собі інформацію та завдання по всім основним кодам, що виявляють помилки, а саме двійкові та недвійкові коди та їх різновидності.

**Вимоги до розробки тренажеру.**

1. Обрати сумісне з системою Moodle середовище розробки.
2. Розробка структурованого алгоритму тренажеру.
3. Розробка блок–схеми алгоритму.
4. Програмна реалізація тренажеру за побудованим алгоритмом та в обраному середовищі розробки.
5. Перевірка правильності роботи всіх елементів тренажеру та внесення останніх правок.
6. Компілювання тренажеру на різні платформи.
7. Завантаження тренажеру до мережі інтернет та в систему Moodle ПУЕТ.

**Критерії щодо розробки програмного продукту.**

1. Розробка простого та зрозумілого інтерфейсу тренажеру.
2. Реалізація різних типів тренування відповідно алгоритму.
3. Підтримка багатомовності. Навчальний тренажер повинен мати підтримку української, російської та англійської мови.
4. Створення скриптів перевірки відповідей користувача, системи підказок та системи виведення правильної відповіді на екран.
5. Реалізація динамічного розрахунку кількості набраних балів та відображення їх користувачу.

6. Реалізація кнопки, яка відкриває сторінку довідки для допомоги у вирішенні поставлених питань.

7. Забезпечення стабільної роботи програми. Виявленням та виправлення всіх можливих помилок .

8. Компіляція програми на три обрані платформи, а саме Android (ver. 5.1 або вище), Windows, HTML5 (web) для забезпечення максимального охоплення користувачів.

# 2 ІНФОРМАЦІЙНИЙ ОГЛЯД

## 2.1. Інформаційний огляд схожих тренажерів

Огляд схожих робіт дає змогу визначити основні переваги та недоліки вже виконаних раніше тренажерів. Для того щоб обрати оптимальні методи взаємодії з користувачем, покращити навички розміщення всіх елементів у тренажері та забезпечити найкращий результат навчання студента було проведено огляд трьох робіт розміщених в системі дистанційного навчання Moodle ПУЕТ .

Першим було розглянуто тренажер Марченка Д. А. «Алгоритмізація та програмна реалізація тренажера з теми «Дослідження на збіжність числових рядів» дистанційного навчального курсу «Математичний аналіз» [5]. Даний тренажер включає в себе, як теоретичні, так і практичні відомості. Користувачу надається вибір розпочати тренування зразу чи після вивчення матеріалу у довідці, де міститься вся потрібна інформація для виконання подальших завдань. При невірній відповіді на питання на екрані відкривається вікно довідки і користувачу надається можливість для повторення та закріплення матеріалу.

Другим було розглянуто тренажер Белінської В. В. «Створення програмного забезпечення тренажера з теми «Розподіли дискретних випадкових величин та їх числові характеристики» дистанційного навчального курсу «Теорія ймовірностей та математична статистика» [6]. Цей тренажер має зручний та зрозумілий інтерфейс, а також можливість обрати одну із трьох доступних мов (українська, російська, англійська). Після невірної відповіді користувачу виводиться на екран вірна відповідь. Доступ до теоретичних відомостей надається через окрему кнопку.

Останньою було розглянуто роботу Самовика С. М. «Розробка алгоритму та програмного забезпечення тренажера з теми «Угорський метод в задачі про призначення» дистанційного навчального курсу «Методи оптимізації та дослідження операцій» [7]. В цьому тренажері також реалізований легкий доступ до інструкції з використання, блок–схеми та загальної інформації про тренажер за допомогою кнопки «Довідка», але немає жодних теоретичних відомостей. Коли

користувач проходить етап тренування та відповідає невірно, на екран виводиться вірна відповідь, але на наступний крок перейти не можна, потрібно замінити свою стару відповідь на нову, що особливо не зручно і займає багато часу. Також відсутнє обмеження на введення тільки цілих цифр.

## 2.2. Переваги оглянутих робіт

Переваги навчального тренажера з теми «Дослідження на збіжність числових рядів» [5]:

1. Перед початком тренування є можливість ознайомитися з теорією.
2. Системна послідовність питань.
3. Зрозумілий інтерфейс тренажеру.
4. Після невірної відповіді надається вся потрібна теоретична інформація для розв’язування задачі, поставленої на даному кроці тренування.

Переваги навчального тренажера з теми «Розподіли дискретних випадкових величин та їх числові характеристики» [6]:

1. Багатомовність тренажеру (доступно 3 мови).
2. Після відповіді користувача відразу відображається вірна відповідь та є можливість перейти на наступний крок.
3. Легкий доступ до теорії на будь–якому етапі.
4. Велика кількість питань, що дає змогу дізнатися та зрозуміти всі аспекти дискретних випадкових величин та їх числових характеристик.

Переваги навчального тренажера з теми «Угорський метод в задачі про призначення» [7]:

1. Продуманість кроків тренажу та різнотипність питань (покрокова демонстрація розв’язання прикладу).
2. Доступ до інструкції з користування тренажером.
3. Інформаційне повідомлення при введенні невірних символів.

## 2.3. Недоліки оглянутих робіт

Основні недоліки оглянутого тренажу Марченка Д. А. [5]:

1. Можливість перейти на наступний крок з’являється тільки після вірної відповіді.
2. Дуже мала кількість кроків.
3. Розглядається тільки теоретична частина теми.
4. Немає розрахунку кінцевого результату проходження тренінгу.
5. Одномовність тренажеру.
6. Для роботи тренажеру потрібно встановити додаткове програмне забезпечення Java.

Недоліки тренажу Белінської В.В. [6]:

1. Використаний не зовсім доцільний шрифт та фон додатку.
2. Після переходу на наступний крок вікно додатку повертається в центр екрану, що не зовсім зручно для користувача.
3. Велика кількість кроків.

Недоліки тренажу Самовика С. М. [7]:

1. Велика складність з введенням відповіді (велика кількість комірок, які потрібно заповнювати на одному кроці).
2. До наступного кроку можливо перейти тільки після введення повністю правильної відповіді, навіть тоді, коли відповідь вже було продемонстровано на екрані.
3. Немає доступу до загальних теоретичних відомостей (лише підказки).
4. Для роботи тренажеру потрібно встановити додаткове програмне забезпечення Java.

Загальні недоліки оглянутих тренажерів:

1. Підтримка лише персональних комп’ютерів, це зумовлено початковим вибором середовища та мови програмування.
2. Застарілий графічний інтерфейс (неможливе масштабування вікна та його компонентів).

3. Наявність незначних граматичних помилок в тестових частинах тренажерів.

4. Немає можливості переглянути попередні відповіді та оцінити кінцевий результат тренування.

## 2.4. Підсумки проведеного інформаційного огляду

Провівши аналіз доступної в мережі інтернет інформації про тренажери з теми «Коди із виявленням помилок» можна зробити висновок, що їх не існує. Переглянуті схожі тренажери вже втратили свою актуальність. Вони мають безліч недоліків та побудовані з використанням застарілих технологій. Тому розробка програмного забезпечення – тренажера дистанційного навчального курсу «Теорія інформації та кодування» з теми «Коди із виявленням помилок» є актуальною для полегшення вивчення, поглиблення та закріплення знань з даної дисципліни в системі дистанційного навчання Moodle.

# 3 ТЕОРЕТИЧНА ЧАСТИНА

## 3.1 Обґрунтування актуальності розробки тренажеру

Враховуючи сучасні тенденції дистанційне навчання набуває все більшої популярності. Більшість університетів України мають свої дистанційні курси за якими навчається значна кількість студентів. Це надає значну гнучкість в системі навчання, як для студента, так і для викладача, адже чудово коли не виходячи з дому можна отримати доступ до будь–якої книги, наукової роботи або тренажеру.

Проводячи аналіз доступних матеріалів в інтернеті та в системі Moodle PUET було знайдено значну кількість неструктурованого матеріалу, в якому студент вивчаючи «Коди із виявленням помилок» міг легко заплутатися, але жодного тренажеру, в якому весь матеріал має структурований вигляд знайдено не було. Враховуючи вищезазначене, створення тренажеру для системи дистанційного навчання Moodle, в якому весь матеріал розділений по темам та легкодоступний, є актуальним.

## 3.2 Алгоритм тренажеру

Після запуску тренажеру перед користувачем з’являється головне меню з назвою тренажеру, вітанням та двома кнопками: «Розпочати» — з’являється меню з кнопками вибору теми тренування та «Вихід» — дає змогу вийти з програми. Всього для тренування доступно 8 тем: «Код з перевіркою на парність», «Код з перевіркою на непарність», «Код із простим повторенням», «Інверсний код (із повторенням та інверсією)», «Кореляційний код», «Код із сталою (постійною) вагою», «Код із кількістю одиниць у комбінації, кратною 3», «Код з перевіркою за модулем q». При виборі будь якої з цих тем користувачу буде надана можливість обрати теоретичну чи практичну частини тестування. Коли користувач повністю завершує проходження однієї з доступних тем, то кнопки з повністю пройденими

темами позначаються зеленим кольором, а кнопки з непройденими темами — червоним.

Коли користувач почне тренування у режимі «Теоретична частина», йому почергово будуть надані тільки тестові питання, кількість яких залежить від обраної теми. При виборі вірної відповіді вона буде відображатися зеленим кольором, при цьому активується кнопка переходу до наступного питання. Коли користувач обирає невірну відповідь, то вона відмічається червоним кольором, демонструється вірна відповідь та з’являється вікно з пропозицією ознайомитися з теоретичним матеріалом за допомогою кнопки «Допомога». Після закінчення тестування користувач зможе перейти до головного меню програми та почати вивчання іншого типу кодів із виявленням помилок, або розпочати виконання практичної частини теми, яку він пройшов.

Якщо користувач почне тренування у режимі «Практична частина», то йому потрібно розв’язати приклад, який складається з декількох кроків, кількість яких залежить від обраної теми. В цьому типі тренування почергово будуть надані питання тільки з введенням відповіді. При введенні вірної відповіді вона буде відображатися зеленим кольором, при цьому активується кнопка переходу до наступного питання. При введені невірної відповіді перший раз – на екрані з’явиться питання – підказка, при введені невірної відповіді другий раз – на екрані з’явиться вірна відповідь, заклик перейти до теорії за допомогою кнопки «Допомога» та з’явиться кнопка переходу до наступного питання.

Розрахунок балів користувача ведеться по кожній темі окремо. За кожну вірну відповідь користувачу нараховується один бал, в залежності від кількості питань максимальна оцінка буде різною. Оцінка буде постійно змінюватися в залежності від правильності відповідей користувача та відображатиметься після завершення одного з етапів теми, біля кнопки обрання теми та динамічно в обраному режимі тренування.

При проходженні теоретичної або практичної частини користувачу весь час будуть доступні такі кнопки: «Завершити» – завершує тестування та переходить до

вікна результатів, «Теоретичний матеріал» – перенаправляє користувача до теоретичного матеріалу, «Відповісти» – програма перевіряє відповідь користувача.

В тексті алгоритму вірні відповіді на теоретичні питання виділені жирним кольором. Відповіді до практичної частини та питання – підказки окремо винесені після кожного практичного питання.

### 3.2.1 Код з перевіркою на парність

***Теоретична частина***

**Крок 1.** Скільки перевірних елементів $(r)$ має код з перевіркою на парність?

A) $r = 2.$

**B)** $\boldsymbol{r = 1.}$

C) $r = -1.$

D) $r = -3.$

**Крок 2.** Скільки інформаційних елементів $(k)$ має код з перевіркою на парність?

**A)** $\boldsymbol{k = (n - 1)}.$

B) $k = (r - 1).$

C) $k = (n - 2).$

D) $k = (n + n - 1).$

**Крок 3.** Як позначається код з перевіркою на парність?

A) $(k, n - 2).$

B) $(r, k - 1).$

**C)** $\boldsymbol{(n, n - 1)}.$

D) $(n, n - 4).$

**Крок 4.** Довжина кодової комбінації $(n)$ обраховується за формулою?

A) $n = k + r - 1.$

B) $n = k - r.$

C) $\boldsymbol{n = k + r.}$

D) $n = k - 2 + r.$

**Крок 5.** Який вигляд має формула для створення перевірного елементу в коді з перевіркою на парність?

**A)** $\boldsymbol{b_1 = \sum_{i=1}^{k} \circ\, a_i}$.

B) $b_1 = \sum_{i=1}^{k-1} \circ\, a_{i+1}$.

C) $b_1 = \sum_{i=1}^{n} \circ\, a_{i+1}$.

D) $b_1 = \sum_{i=1}^{n} \circ\, a_{i-1}$.

**Крок 6.** Як утворюється кодова комбінація в коді з перевіркою на парність?

**A) Кодова комбінація утворюється доповненням комбінації $k$ – елементного первинного коду одним елементом таким чином, щоб кількість одиниць у новому n – розрядному ( $n = k + 1$) коді була парною.**

B) Кожна комбінація має непарну кількість одиниць, тобто додатковий перевірний елемент формують, виходячи з кількості одиниць у початковій кодовій комбінації; при парній кількості перевірний елемент дорівнює одиниці, а при непарній – нулю.

C) Кодова комбінація формується з $r$ перевірних елементів які є простим повторенням $k$ інформаційних елементів первинної кодової комбінації $b_i = a_i$, де $i = 1 \ldots k$.

**Крок 7.** Яку мінімальну кодову відстань має код з перевіркою на парність $d_{min}$ =?

A) $d_{min} = 1$.

B) $d_{min} = 3$.

C) $d_{min} = n + 1$.

**D)** $\boldsymbol{d_{min} = 2}$.

**Крок 8.** Який кодовий синдром визначають перевіряючи усю прийняту комбінацію на парність де $a_i'$, $b_i'$ – прийняті на приймальному боці відповідно інформаційні та перевірні елементи.

A) $S = \sum_{i=1}^{n-2} \circ\, a_i' \oplus b_1'$.

B) $S = \sum_{i=1}^{n-1} \circ\, a_i + b_1$.

**C)** $\boldsymbol{S = \sum_{i=1}^{n-1} \circ\, a_i' \oplus b_1'}$.

D) $S = \sum_{i=1}^{n-1} \circ a'_i - b'_1$.

**Крок 9.** Про відсутність помилки в комбінації свідчить умова?

**A)** $\boldsymbol{S = 0}$.

B) $S = 1$.

C) $S = -1$.

D) $S = 2$.

**Крок 10.** Яким виразом визначається надмірність коду з перевіркою на парність?

A) $R_{над} = 1 - k/(k+2)$.

B) $R_{над} = 1 - (log_2 C_n^m)/n$.

**C)** $\boldsymbol{R_{над} = 1 - k/(k+1) = 1/(k+1)}$.

D) $R_{над} = 1 - k/(2k) = \frac{1}{2}$.

***Практична частина***

**Приклад.** Закодувати комбінацію 0110110 двійкового простого коду ($k = 7$) двійковими кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити надмірність коду.

**Крок 1.** Побудуйте кодову комбінацію кодом з перевіркою на парність.

***Відповідь:*** $A = 01101100$.

**Питання – підказка:** Крок 5 та 6 з теоретичної частини даної теми.

**Крок 2.** Розрахуйте довжину кодової комбінації ($n =$).

***Відповідь:*** $n = k + r = 8$.

**Питання – підказка:** Крок 4 з теоретичної частини даної теми.

**Крок 3.** Нехай виникла однократна помилка, вектор якої $E = 00100000$. Тоді кодова комбінація на приймальному боці матиме вигляд ($A' = A \oplus E =$).

***Відповідь:*** $A' = 01001100$.

**Крок 4.** Розрахуйте кодовий синдром ($S =$ ).

***Відповідь:*** $S = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 = 1$.

**Питання – підказка:** Крок 8 з теоретичної частини даної теми.

**Крок 5.** Використавши результат обрахунку кодового синдрому $S$ оберіть правильну відповідь.

**A) Існує помилка.**

B) Помилка відсутня.

**Питання – підказка:** Крок 9 з теоретичної частини даної теми.

**Крок 6.** Порахуйте надмірність коду ($R_{\text{над}} =$).

***Відповідь:*** $R_{\text{над}} = 1 - \frac{7}{8} = \ 0{,}125.$

**Питання – підказка:** Крок 10 з теоретичної частини даної теми.

### 3.2.2 Код з перевіркою на непарність

***Теоретична частина***

**Крок 1.** Скільки перевірних елементів має код з перевіркою на непарність ($r =$) ?

A) $\mathrm{r} = 2.$

B) $\mathrm{r} = -1$

C) $\mathbf{r = 1.}$

D) $\mathrm{r} = -3.$

**Крок 2.** Скільки інформаційних елементів має код з перевіркою на непарність ($k =$) ?

A) $\boldsymbol{k = (n - 1)}.$

B) $k = (r - 1).$

C) $k = (n - 2).$

D) $k = (n + n - 1).$

**Крок 3.** Як позначається код з перевіркою на непарність?

A) $(k, n - 2).$

B) $(r, k - 1).$

**C)** $\boldsymbol{(n, n - 1).}$

D) $(n, n-4)$.

**Крок 4.** Який вигляд має формула для створення перевірного елементу в коді з перевіркою на непарність?

**A)** $\boldsymbol{b_1 = \sum_{i=1}^{k} \circ\, a_i}$.

B) $b_1 = \sum_{i=1}^{k-1} \circ\, a_{i+1}$.

C) $b_1 = \sum_{i=1}^{n} \circ\, a_{i+1}$.

D) $b_1 = \sum_{i=1}^{n} \circ\, a_{i-1}$.

**Крок 5.** Як утворюється кодова комбінація в коді з перевіркою на непарність?

A) Кодова комбінація утворюється доповненням комбінації k – елементного первинного коду одним елементом таким чином, щоб кількість одиниць у новому n – розрядному ($n = k + 1$) коді була парною.

**B) Кожна комбінація має непарну кількість одиниць, тобто додатковий перевірний елемент формують, виходячи з кількості одиниць у початковій кодовій комбінації; при парній кількості перевірний елемент дорівнює одиниці, а при непарній – нулю.**

C) Кодова комбінація формується з $r$ перевірних елементів які є простим повторенням $k$ інформаційних елементів первинної кодової комбінації $b_i = a_i$, де $i = 1 \ldots k$.

**Крок 6.** Яку мінімальну кодову відстань має код з перевіркою на непарність $d_{min}$ =?

A) $d_{min} = 1$.

B) $d_{min} = 3$.

C) $d_{min} = n + 1$.

**D)** $\boldsymbol{d_{min} = 2}$.

**Крок 7.** Який кодовий синдром визначають перевіряючи усю прийняту комбінацію на парність де $a_i'$, $b_i'$ – прийняті на приймальному боці відповідно інформаційні та перевірні елементи.

A) $S = \sum_{i=1}^{n-2} \circ\, a_i' \oplus b_1'$.

B) $S = \sum_{i=1}^{n-1} \circ a'_1 + b'_i.$

**C)** $\mathbf{S = \sum_{i=1}^{n-1} \circ a'_i \oplus b'_1.}$

D) $S = \sum_{i=1}^{n-1} \circ a'_i - b'_1.$

**Крок 8.** Про відсутність помилки в комбінації свідчить умова?

A) $S = 0.$

**B)** $\boldsymbol{S = 1.}$

C) $S = 2.$

D) $S = 3.$

**Крок 9.** Яким виразом визначається надмірність коду з перевіркою на непарність?

A) $R_{над} = 1 - (log_2 C_n^m)/n.$

B) $R_{над} = 1 - k/(k+2).$

**C)** $\boldsymbol{R_{над} = 1 - k/(k+1) = 1/(k+1).}$

D) $R_{над} = 1 - k/(2k) = \frac{1}{2}.$

***Практична частина***

**Приклад.** Закодувати комбінацію 0110110 двійкового простого коду ($k = 7$) двійковими кодами, що виявляють помилки з перевіркою на непарність, виявити однократну помилку та визначити надмірність коду.

**Крок 1.** Побудуйте кодову комбінацію кодом з перевіркою на непарність.

***Відповідь:*** $A = 01101101.$

**Питання – підказка:** Крок 5 з теоретичної частини даної теми.

**Крок 2.** Розрахуйте довжину кодової комбінації ($n =$).

***Відповідь:*** $n = k + r = 8.$

**Питання – підказка:** Крок 4 з теоретичної частини теми «Код з перевіркою на парність».

**Крок 3.** Нехай виникла однократна помилка, вектор якої $E = 10000000.$ Тоді кодова комбінація на приймальному боці матиме вигляд($A' = A \oplus E =$).

***Відповідь:*** $A' = 11101101.$

**Крок 4.** Розрахуйте кодовий синдром ( $S =$).

***Відповідь:*** $S = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 = 0.$

**Питання – підказка:** Крок 7 з теоретичної частини даної теми.

**Крок 5.** Використавши результат обрахунку кодового синдрому $S$ оберіть правильну відповідь.

**A) Існує помилка.**

B) Помилка відсутня.

**Питання – підказка:** Крок 8 з теоретичної частини даної теми.

**Крок 6.** Порахуйте надмірність коду ($R_{\text{над}} =$).

***Відповідь:*** $R_{\text{над}} = 1 - \frac{7}{8} = \ 0{,}125.$

**Питання – підказка:** Крок 9 з теоретичної частини даної теми.

### 3.2.3 Код із простим повторенням

***Теоретична частина***

**Крок 1.** Скільки перевірних елементів має код із простим повторенням ($r =$) ?

**A)** $\boldsymbol{r = k.}$

B) $r = k + 1.$

C) $r = k - 1.$

D) $r = 1.$

**Крок 2.** Як утворюється кодова комбінація в коді з простим повторенням?

A) Кодова комбінація утворюється доповненням комбінації k – елементного первинного коду одним елементом таким чином, щоб кількість одиниць у новому n – розрядному ($\mathrm{n} = \mathrm{k} + 1$) коді була парною.

B) Кожна комбінація має непарну кількість одиниць, тобто додатковий перевірний елемент формують, виходячи з кількості одиниць у початковій кодовій комбінації; при парній кількості перевірний елемент дорівнює одиниці, а при непарній – нулю.

**C) Кодова комбінація формується з r перевірних елементів які є простим повторенням k інформаційних елементів первинної кодової комбінації $b_i = a_i$, де $i = 1 \dots k$.**

**Крок 3.** Яку мінімальну кодову відстань має код із простим повторенням $d_{min} = ?$

A) $d_{min} = 1.$

**B)** $\boldsymbol{d_{min} = 2.}$

C) $d_{min} = n + 1.$

D) $d_{min} = 4.$

**Крок 4.** Яким методом перевіряють наявність помилок у закодованій комбінації кодом із простим повторенням?

**A) Потрібно провести порівняння однойменних інформаційних і перевірних елементів у прийнятій кодовій комбінації. Незбіг їх свідчить про наявність помилок у ній.**

B) Декодування кодової комбінації у декодері ведуть тактами по два елементи у кожному такті. При цьому два елементи одного такту не повинні мати однакове значення, тобто не повинно бути сполучень «00» та «11».

**Крок 5.** Яким виразом визначається надмірність коду із простим повторенням?

A) $R_{\text{над}} = 1 - [log_2(C_n^0 + C_n^3 + C_n^6 + \cdots + C_n^{3b})] \backslash n.$

B) $R_{\text{над}} = 1 - (log_2 C_n^m)/n.$

C) $R_{\text{над}} = 1 - k/(k+1) = 1/(k+1).$

**D)** $\boldsymbol{R_{\text{над}} = 1 - k/(2k) = \frac{1}{2}.}$

***Практична частина***

**Приклад.** Закодувати комбінацію 0110110 двійкового простого коду ($k = 7$) двійковими кодами двійковими кодами з простим повторенням, виявити однократну помилку та визначити надмірність коду.

**Крок 1.** Закодуйте двійкову комбінацію кодом з простим повторенням.

***Відповідь:*** $A = 01101100110110.$

**Питання – підказка:** Крок 2 з теоретичної частини даної теми.

**Крок 2.** Розрахуйте довжину кодової комбінації для заданої в прикладі двійкової комбінації ($n =$).

***Відповідь:*** $n = k + r = 14.$

**Питання – підказка:** Крок 4 з теоретичної частини теми «Код з перевіркою на парність».

**Крок 3.** Нехай виникла однократна помилка, вектор якої $E = 00000010000000$. Тоді кодова комбінація на приймальному боці матиме вигляд ($A' = A \oplus E =$).

***Відповідь:*** $A' = 01101110110110.$

**Крок 4.** Перевірте $A'$ на наявність помилки.

**A) Існує помилка.**

B) Помилка відсутня.

**Питання – підказка:** Крок 4 з теоретичної частини теми даної теми.

**Крок 5.** Порахуйте надмірність коду ($R_{\text{над}} =$).

***Відповідь:*** $R_{\text{над}} = 1 - \frac{7}{2*7} = \frac{1}{2}.$

**Питання – підказка:** Крок 5 з теоретичної частини теми даної теми.

### 3.2.4 Інверсний код (із повторенням та інверсією)

#### *Теоретична частина*

**Крок 1.** Скільки перевірних елементів має інверсний код ($r =$) ?

A) $r = 1.$

B) $r = k + 1.$

C) $r = k - 1.$

**D)** $\boldsymbol{r = k.}$

**Крок 2.** За умови $\sum_{i=1}^{k} \circ a_i = 0$, тобто при парній кількості одиниць у початковій кодовій комбінації, перевірні елементи?

**A) Просто повторюють інформаційні ( $\boldsymbol{b_i = a_i, i = 1, \dots, k}$).**

B) Повторюють інформаційні в інвертованому вигляді $b_i = a_i \oplus 1, i = 1, \dots, k$.

**Крок 3.** За умови $\sum_{i=1}^{k} \circ a_i = 1$, тобто при непарній кількості одиниць у початковій кодовій комбінації, перевірні елементи?

A) Просто повторюють інформаційні ( $b_i = a_i, i = 1, \dots, k$).

**B) Повторюють інформаційні в інвертованому вигляді ($\boldsymbol{b_i = a_i \oplus 1, i = 1, \dots, k}$).**

**Крок 4.** Яку мінімальну кодову відстань має інверсний код $d_{min}$ =?

A) $d_{min} = 1.$

B) $d_{min} = 3.$

**C) $\boldsymbol{d_{min} = 2.}$**

D) $d_{min} = n + 1.$

**Крок 5.** Як виявляють наявність помилок на приймальному боці у послідовності?

A) Потрібно провести порівняння однойменних інформаційних і перевірних елементів у прийнятій кодовій комбінації. Незбіг їх свідчить про наявність помилок у ній.

**B) Спочатку підсумовують одиниці, які знаходяться в перших k елементах. Якщо їх кількість парна, то решту k елементів приймають у позитиві. Обидві зареєстровані частини комбінацій поелементно порівнюють. За наявності хоча б одного незбігу вся послідовність елементів бракується. Якщо кількість одиниць серед перших k елементів непарна, то решту k елементів приймають у негативі (інвертують), після чого поелементно порівнюють їх. Наявність незбігу призводить до відбраковування всіх 2k елементів.**

C) Декодування кодової комбінації у декодері ведуть тактами по два елементи у кожному такті. При цьому два елементи одного такту не повинні мати однакове значення, тобто не повинно бути сполучень «00» та «11».

**Крок 6.** Яким виразом визначається надмірність інверсного коду ?

A) $R_{\text{над}} = 1 - \frac{k}{(k+2)}$.

B) $R_{\text{над}} = 1 - (log_2 C_n^m)/n.$

C) $R_{\text{над}} = 1 - k/(k+1) = 1/(k+1).$

**D)** $\boldsymbol{R_{\text{над}} = 1 - k/(2k) = \frac{1}{2}.}$

***Практична частина***

**Приклад.** Закодувати комбінацію 110010 двійкового простого коду ($k = 6$) інверсним кодом (із повторенням та інверсією), виявити однократну помилку та визначити надмірність коду.

**Крок 1.** Закодуйте двійкову комбінацію інверсним кодом.

***Відповідь:*** $A = 110010001101$ .

**Питання – підказка:** Крок 2 та 3 з теоретичної частини даної теми.

**Крок 2.** Розрахуйте довжину кодової комбінації ($n =$).

***Відповідь:*** $n = k + r \ = \ 12.$

**Питання – підказка:** Крок 4 з теоретичної частини теми «Код з перевіркою на парність».

**Крок 3.** Нехай виникла однократна помилка, вектор якої $E =$ 000010000000. Тоді кодова комбінація на приймальному боці матиме вигляд ($A' = A \oplus E =$).

***Відповідь:*** $A' = 110000001101.$

**Крок 4.** Перевірте $A'$ на наявність помилки.

**A) Існує помилка.**

B) Помилка відсутня.

**Питання – підказка:** Крок 5 з теоретичної частини даної теми.

**Крок 5.** Порахуйте надмірність коду ($R_{\text{над}} =$).

***Відповідь:*** $R_{\text{над}} = 1 - \frac{6}{2*6} = \frac{1}{2}$.

**Питання – підказка:** Крок 6 з теоретичної частини даної теми.

### 3.2.5 Кореляційний код

***Теоретична частина***

**Крок 1.** Скільки перевірних елементів має кореляційний код ($r =$) ?

**A)** $\boldsymbol{r = k.}$

B) $r = k + 1.$

C) $r = k - 1.$

D) $r = 1.$

**Крок 2.** При кодуванні інформації «0» записується як?

A) «00».

**B)** «**01**».

C) «10».

D) «10».

**Крок 3.** При кодуванні інформації «1» записується як?

A) «00».

B) «01».

**C)** «**10**».

D) «10».

**Крок 4.** Яку мінімальну кодову відстань має кореляційний код $d_{min} =$?

A) $d_{min} = 1.$

B) $d_{min} = 3.$

**C)** $\boldsymbol{d_{min} = 2.}$

D) $d_{min} = n + 1.$

**Крок 5.** Як виявляють наявність помилок на приймальному боці у послідовності?

A) Потрібно провести порівняння однойменних інформаційних і перевірних елементів у прийнятій кодовій комбінації. Незбіг їх свідчить про наявність помилок у ній.

**B) Декодування кодової комбінації у декодері ведуть тактами по два елементи у кожному такті. При цьому два елементи одного такту не повинні мати однакове значення, тобто не повинно бути сполучень «00» та «11».**

C) До первинної кодової комбінації додаються два перевірних розряди, які мають такі значення, що сума одиниць у кодовій комбінації стає кратною трьом.

**Крок 6.** Яким виразом визначається надмірність кореляційного коду ?

A) $R_{\text{над}} = 1 - (log_2 C_n^m)/n.$

**B)** $\boldsymbol{R_{\text{над}} = 1 - k/(2k) = 1/2.}$

C) $R_{\text{над}} = 1 - k/(k+1) = 1/(k+1).$

D) $R_{\text{над}} = 1 - [log_2(C_n^0 + C_n^3 + C_n^6 + \cdots + C_n^{3b})]\backslash n.$

***Практична частина***

**Приклад.** Закодувати комбінацію 010011 двійкового простого коду ($k = 6$) кореляційним кодом, виявити однократну помилку та визначити надмірність коду.

**Крок 1.** Закодуйте двійкову комбінацію кореляційним кодом.

***Відповідь:*** $A =$ 011001011010.

**Питання – підказка:** Крок 2 та 3 з теоретичної частини даної теми.

**Крок 2.** Розрахуйте довжину кодової комбінації (n=).

***Відповідь:*** $n = k + r = 12.$

**Питання – підказка:** Крок 4 з теоретичної частини теми «Код з перевіркою на парність».

**Крок 3.** Нехай виникла однократна помилка, вектор якої $E =$ 000010000000. Тоді кодова комбінація на приймальному боці матиме вигляд ($A' = A \oplus E =$)?

***Відповідь:*** $A' = 01100011010.$

**Крок 4.** Перевірте $A'$ на наявність помилки.

**A) Існує помилка.**

B) Помилка відсутня.

**Питання – підказка:** Крок 5 з теоретичної частини даної теми.

**Крок 5.** Порахуйте надмірність коду ($R_{\text{над}} =$).

***Відповідь:*** $R_{\text{над}} = 1 - \frac{6}{2*6} = \frac{1}{2}.$

**Питання – підказка:** Крок 6 з теоретичної частини даної теми.

### 3.2.6 Код із сталою (постійною) вагою

***Теоретична частина***

**Крок 1.** Яку мінімальну кодову відстань має код із сталою вагою ($d_{min} =$)?

A) $d_{min} = 1.$

B) $d_{min} = 3.$

**C)** $\boldsymbol{d_{min} = 2.}$

D) $d_{min} = n + 1.$

**Крок 2.** Загальна кількість кодових комбінацій коду з постійною вагою $C_n^m$, де параметром m позначають?

A) Довжину комбінації.

B) Кількість помилок.

**C) Кількість одиниць у кодовій комбінації.**

D) Загальну кількість кодових комбінацій.

**Крок 3.** Загальна кількість кодових комбінацій коду з постійною вагою $C_n^m$, де параметром $n$ позначають?

**A) Довжину кодової комбінації.**

B) Кількість помилок.

C) Кількість одиниць у кодовій комбінації.

D) Загальну кількість кодових комбінацій.

**Крок 4.** Загальна кількість кодових комбінацій коду з постійною вагою ($N =$) обчислюється за формулою?

**A)** $\boldsymbol{N = C_n^m = \frac{n!}{m!(n-m)!}}$.

B) $N = C_m^n = \frac{m!}{n!(m-n)!}$.

C) $N = C_k^m = \frac{k!}{m(k+m)!}$.

**Крок 5.** Як виявляють наявність помилок на приймальному боці у отриманій послідовності?

A) До первинної кодової комбінації додаються два перевірних розряди, які мають такі значення, що сума одиниць у кодовій комбінації стає кратною трьом.

B) Потрібно провести порівняння однойменних інформаційних і перевірних елементів у прийнятій кодовій комбінації. Незбіг їх свідчить про наявність помилок у ній.

**C) Код утворюється з простого двійкового коду відбором комбінацій, які мають однакову кількість одиниць $m$. У декодері підраховується кількість одиниць у прийнятій кодовій комбінації. Невідповідність кількості одиниць числу m говорить про наявність помилки у кодовій комбінації.**

D) Декодування кодової комбінації у декодері ведуть тактами по два елементи у кожному такті. При цьому два елементи одного такту не повинні мати однакове значення, тобто не повинно бути сполучень «00» та «11».

**Крок 6.** Код із сталою вагою виявляє помилки.

A) Всі помилки непарної кратності.

B) Всі помилки парної кратності, які призводять до порушення умови $m = const$.

**C) Помилки у варіантах $a$ та $b$.**

**Крок 7.** Яким виразом визначається надмірність коду із сталою вагою?

**A)** $\boldsymbol{R_{\text{над}} = 1 - (log_2 C_n^m)/n}$.

B) $R_{\text{над}} = 1 - k/(2k) = 1/2.$

C) $R_{\text{над}} = 1 - k/(k+1) = 1/(k+1).$

D) $R_{\text{над}} = 1 - k/(k+2).$

***Практична частина***

**Приклад.** Перевірити отримані комбінації $A_1 = 0100011$ та $A_2 = 0101011$ двійкового простого коду $n = 6$ кодом зі сталою вагою на наявність помилки, визначити загальну кількість кодових комбінацій та надмірність коду. Число одиниць у комбінації $m = 3$.

**Крок 1.** Перевірити комбінацію $A_1 = 010011$ на наявність помилки $m = 3$.

A) Існує помилка.

**B) Помилка відсутня.**

**Питання – підказка:** Крок 2 та 5 з теоретичної частини даної теми.

**Крок 2.** Перевірити комбінацію $A_2 = 010111$ на наявність помилки $m = 3$.

**A) Існує помилка.**

B) Помилка відсутня.

**Питання – підказка:** Крок 2 та 5 з теоретичної частини даної теми.

**Крок 3.** Розрахуйте загальну кількість кодових комбінацій коду з постійною вагою ($N =$) коли $n = 7, m = 3$.

***Відповідь:*** $N = C_n^m = \frac{n!}{m!(n-m)!} = 35.$

**Питання – підказка:** Крок 4 з теоретичної частини даної теми.

**Крок 4.** Порахуйте надмірність коду ($R_{\text{над}} =$).

***Відповідь:*** $R_{\text{над}} = 1 - \frac{log_2 35}{7} = 0{,}267.$

**Питання – підказка:** Крок 7 з теоретичної частини даної теми.

### 3.2.7 Код із кількістю одиниць у комбінації, кратною 3

***Теоретична частина***

**Крок 1.** Код із кількістю одиниць у комбінації, кратною 3 можна утворити.

A) Шляхом додавання до кожної комбінації первинного коду двох перевірних елементів (група 1).

B) Шляхом зменшення кількості дозволених кодових комбінацій первинного коду за допомогою накладання додаткової умови – кількість одиниць у кожній комбінації повинна бути кратною трьом (група 2).

**C) Код можна утворити як шляхам $a$ (група 1) так і $b$ (група 2).**

**Крок 2.** Скільки перевірних елементів має код із кількістю одиниць у комбінації, кратною 3 (групи 1) ($r =$) ?

A) $r = k.$

B) $r = k + 1.$

**C) $r = 2.$**

D) $r = 1.$

**Крок 3.** Параметром $w$ позначають?

A) Цілу частину від $n\backslash 3.$

B) Кількість помилок.

**C) Кількість одиниць у кодовій комбінації (вага).**

D) Загальну кількість кодових комбінацій.

**Крок 4.** Як виявляють наявність помилок на приймальному боці у отриманій послідовності утвореної методом групи 1?

A) Потрібно провести порівняння однойменних інформаційних і перевірних елементів у прийнятій кодовій комбінації. Незбіг їх свідчить про наявність помилок у ній.

**B) До первинної кодової комбінації додаються два перевірних розряди, які мають такі значення, що сума одиниць у кодовій комбінації стає кратною трьом.**

C) з усіх кодових комбінацій первинного коду вибирають тільки ті комбінації, які мають вагу w = 0, 3, 6, 9, . . . Всі інші комбінації заборонені для вживання.

D) Декодування кодової комбінації у декодері ведуть тактами по два елементи у кожному такті. При цьому два елементи одного такту не повинні мати однакове значення, тобто не повинно бути сполучень «00» та «11».

**Крок 5.** Як виявляють наявність помилок на приймальному боці у отриманій послідовності утвореної методом групи 2?

A) Потрібно провести порівняння однойменних інформаційних і перевірних елементів у прийнятій кодовій комбінації. Незбіг їх свідчить про наявність помилок у ній.

B) До первинної кодової комбінації додаються два перевірних розряди, які мають такі значення, що сума одиниць у кодовій комбінації стає кратною трьом.

**C) З усіх кодових комбінацій первинного коду вибирають тільки ті комбінації, які мають вагу $w = 0, 3, 6, 9, \ldots$ Всі інші комбінації заборонені для вживання.**

D) Декодування кодової комбінації у декодері ведуть тактами по два елементи у кожному такті. При цьому два елементи одного такту не повинні мати однакове значення, тобто не повинно бути сполучень «00» та «11».

**Крок 6.** Яким виразом визначається надмірність коду (групи 1)?

A) $R_{\text{над}} = 1 - (log_2 C_n^m)/n.$

**B)** $\boldsymbol{R_{\text{над}} = 1 - k/(k+2).}$

C) $R_{\text{над}} = 1 - k/(2k) = 1/2.$

D) $R_{\text{над}} = 1 - k/(k+1) = 1/(k+1).$

**Крок 7.** Яким виразом визначається надмірність коду (групи 2)?

A) $R_{\text{над}} = 1 - (log_2 C_n^m)/n.$

B) $R_{\text{над}} = 1 - k/(k+2).$

C) $R_{\text{над}} = 1 - k/(2k) = 1/2.$

**D)** $\boldsymbol{R_{\text{над}} = 1 - [log_2(C_n^0 + C_n^3 + C_n^6 + \cdots + C_n^{3b})] \backslash n.}$

**Крок 8.** Яке значення позначають параметром $b$ у формулі розрахунку надмірності коду в якому кількість одиниць кратна трьом?

**A) Цілу частину від $\boldsymbol{n \backslash 3}$.**

B) Кількість одиниць у кодовій комбінації.

C) Довжину комбінації.

D) Загальну кількість кодових комбінацій.

***Практична частина***

**Приклад.** Закодувати комбінацію $000011$ двійкового простого коду ($k = 6$) кодом із кількістю одиниць у комбінації, кратною 3 (група 1), виявити однократну помилку, визначити надмірність коду та перевірити можливість використання групи 2.

**Крок 1.** Закодувати комбінацію $000011$ кодом із кількістю одиниць у комбінації, кратною 3 (група 1).

***Відповідь:*** $A = 00001101$.

**Питання – підказка:** Крок 1 з теоретичної частини даної теми.

**Крок 2.** Розрахуйте довжину кодової комбінації (n=) для групи 1.

***Відповідь:*** $n = k + r = 8$.

**Питання – підказка:** Крок 4 з теоретичної частини теми «Код з перевіркою на парність».

**Крок 3.** Нехай виникла однократна помилка, вектор якої $E = 01000000$. Тоді кодова комбінація на приймальному боці матиме вигляд ($A' = A \oplus E =$).

***Відповідь:*** $A' = 01001101$.

**Крок 4.** Перевірте $A'$ на наявність помилки (група 1).

**A) Існує помилка.**

B) Помилки не існує.

**Питання – підказка:** Крок 4 з теоретичної частини даної теми.

**Крок 5.** Чи можливо використати код із кількістю одиниць у комбінації, кратною 3 (група 2) для кодування комбінації 000011.

**A) Ні.**

B) Так.

**Питання – підказка:** Крок 1**,** Крок 3 та 5 з теоретичної частини даної теми.

**Крок 6.** Порахуйте надмірність коду ($R_{\text{над}} =$) для групи 1.

***Відповідь:*** $R_{\text{над}} = 1 - \frac{6}{(6+2)} = \frac{1}{4}$.

**Питання – підказка:** Крок 7 з теоретичної частини даної теми.

**Крок 7.** Порахуйте надмірність коду ($R_{\text{над}} =$) для групи 2.

***Відповідь:*** $R_{\text{над}} = 1 - [log_2(C_6^0 + C_6^3)] \backslash n = 0{,}450$

**Питання – підказка:** Крок 8 з теоретичної частини даної теми.

### 3.2.8 Код з перевіркою за модулем q

***Теоретична частина***

**Крок 1.** Код з перевіркою за модулем q будується за аналогією.

**A) Двійковим кодом з перевіркою на парність.**

B) Кодом із сталою (постійною) вагою.

C) Кодом із простим повторенням.

**Крок 2.** Скільки перевірних елементів ($r$) має код з перевіркою за модулем q?

A) 2.

**B) 1.**

C) 0.

D) Не містить перевірних елементів.

**Крок 3.** Код з перевіркою за модулем q можна утворити.

A) Шляхом додавання до кожної комбінації первинного коду двох перевірних елементів (група 1).

**B) Доповненням кодової комбінації первинного q коду перевірним елементом таким чином, щоб сума усіх елементів дорівнювала нулю за mod q.**

C) Доповненням комбінації k – елементного первинного коду одним елементом таким чином, щоб кількість одиниць у новому n – розрядному (n = k + 1) коді була парною.

**Крок 4.** Значення перевірного елемента ($b_1$) у кодах з перевіркою за модулем q визначається?

A) $b_1 = (a_0 \oplus a_1 \oplus a_2 \oplus \ldots \oplus a_{k+1}) mod q.$

B) $b_1 = (a_1 \oplus a_2 \oplus \ldots \oplus a_n) mod q.$

C) $b_1 = (a_1 \oplus a_2 \oplus \ldots \oplus a_{k+n}) mod q.$

**D)** $\boldsymbol{b_1 = (a_1 \oplus a_2 \oplus \ldots \oplus a_k) mod q.}$

**Крок 5.** Як виявляють наявність помилок на приймальному боці у отриманій послідовності утвореної кодом з перевіркою за модулем q?

A) Якщо сума усіх елементів (інформаційних та перевірних) за $mod q$ дорівнює нулю.

**B) Якщо сума усіх елементів (інформаційних та перевірних) за $\boldsymbol{mod q}$ відрізняється від нуля.**

C) Якщо сума усіх інформаційних елементів за $mod q$ відрізняється від нуля.

D) Якщо сума усіх інформаційних елементів за $mod q$ дорівнює нулю.

**Крок 6.** Яким виразом визначається надмірність коду з перевіркою за модулем q?

A) $R_{\text{над}} = 1 - (log_2 C_n^m)/n.$

B) $R_{\text{над}} = 1 - k/(2k) = 1/2.$

**C)** $\boldsymbol{R_{\text{над}} = 1/(k+1).}$

D) $R_{\text{над}} = 1 - k/(k+2).$

***Практична частина***

**Приклад.** Закодувати комбінацію 102 трійкового коду недвійковими кодами що виявляють помилки за модулем $q = 3$, виявити помилку, визначити надмірність коду.

**Крок 1.** Закодуйте комбінацію 102 недвійковим кодом за модулем $q = 3$.

***Відповідь:*** $A = 1020$.

**Питання – підказка:** Крок 3 та 4 з теоретичної частини даної теми.

**Крок 2.** Нехай виникла помилка, вектор якої $E = 0100$. Тоді кодова комбінація на приймальному боці матиме вигляд ($A' = A \oplus E =$).

***Відповідь:*** $A' = 1120$.

**Крок 3.** Перевірте $A'$ на наявність помилки.

**А) Існує помилка.**

В) Помилка відсутня.

**Питання – підказка:** Крок 5 з теоретичної частини даної теми.

**Крок 4.** Порахуйте надмірність коду ($R_{\text{над}} =$).

***Відповідь:*** $R_{\text{над}} = \frac{1}{3+1} = 0{,}25$.

**Питання – підказка:** Крок 6 з теоретичної частини даної теми.

## 3.3 Блок–схема алгоритму тренажера

Для розуміння структури тренажеру побудовано його блок–схему, яку зображено на рис. А.1 – А.3. На блок–схемі зображено роботу тренажеру з розділеними на типи питаннями:

- тестові питання — питання з можливістю вибору відповіді (в блок–схемі розпочинається із розриву «А», зображено на рис. А.2);
- питання з введенням відповіді (в блок–схемі розпочинається із розриву «Б», зображено на рис. А.3);

Також представлено основну частину блок–схеми, яка містить в собі основний алгоритм тренажеру (рис. А.1).

# 4 ПРАКТИЧНА ЧАСТИНА

## 4.1 Опис коду тренажеру

Тренажер розроблено за допомогою середовища розробки додатків Unity [8], об'єктно–орієнтовної мови програмування C# та редактору коду Microsoft Visual Studio.

Тренажер складається з двох сцен:

- «StartScene» (рис. 4.1) — складається з вікна привітання та трьох кнопок «Мова»– вибір мови, «Почати» – починає тренування, «Вихід» – закриває вікно тренажеру.
- «MainScene» (рис. 4.2) — містить в собі всі інші елементи тренажеру, на цій сцені відображається головне меню, вікно тренування та вікно з повідомленням про проходження тренінгу. Кнопки відповіді, обрання режиму тренування, виходу та допомоги.

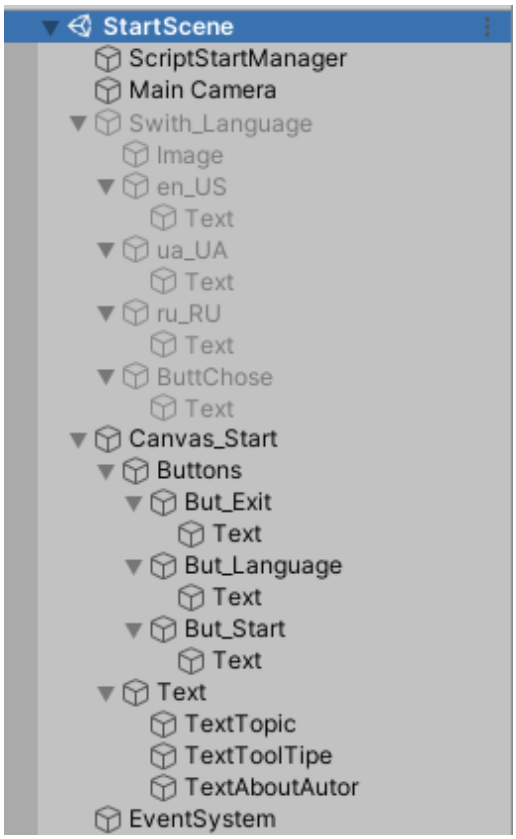


Рисунок 4.1 — Ієрархія сцени «StartScene» у Unity

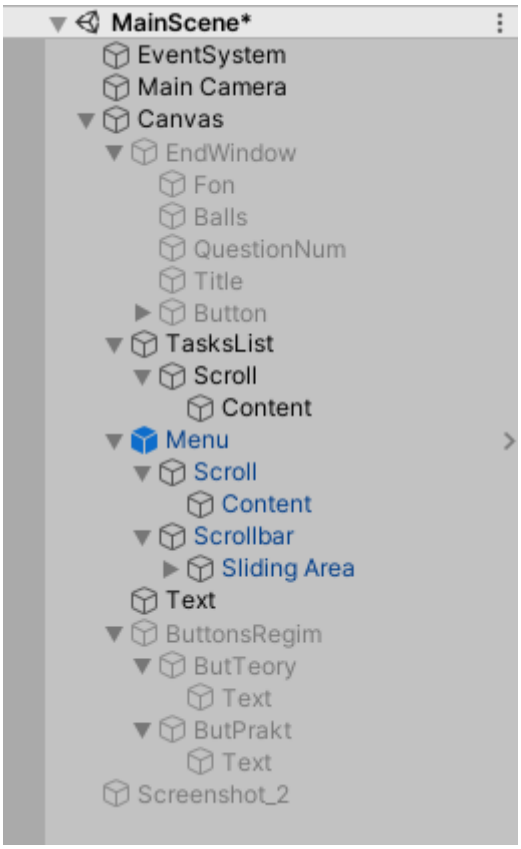


Рисунок 4.2 — Ієрархія сцени «MainScene» у Unity

Для того щоб не створювати велику кількість схожих вікон самостійно було створено префаби – шаблони, в які динамічно за допомогою скриптів завантажується інформація. За допомогою префабів створені вікна тренування, меню, завершення тестування та деякі компоненти, які динамічно змінюють своє значення. На рис 4.3 зображено префаби з використанням яких забезпечується робота тренажеру.

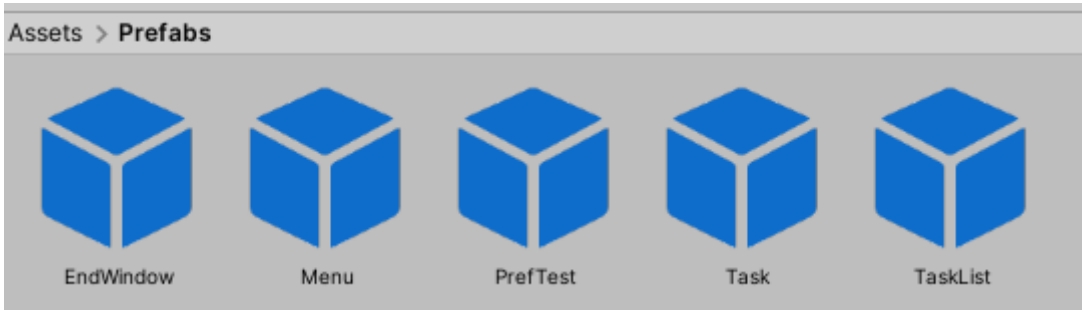


Рисунок 4.3 — Створені префаби

Програма складається з великої кількості об’єктів, серед яких найчастіше використовувалися такі: «Text», «Button», «Toggle», «Canvas», «Camera», «Input Field» та написаних на мові програмування C# чотирнадцяти скриптів, які відповідають за роботу програми, обробку вибраних користувачем відповідей та перехід між питаннями. Код скриптів представлено в Додатку Б.

Для забезпечення багатомовного інтерфейсу у тренажері використовуються файли з розширенням «.json». Створено три файли (рис. 4.4) «ua_UA.json» – українська мова, «ru_RU.json» – російська мова, «en_US.json»– англійська мова, вміст яких відображено в пункті (Б.1 – Б.3). В цих файлах міститься весь текст, що відображається в тренажері. Переключення між мовами та зчитування даних з цих файлів забезпечується за допомогою скриптів: «BtnSwitchLang»– забезпечує роботу кнопок зміни мови (Б.4), «LocalizationData» – клас в якому зберігаються зчитані з файлів дані (Б.5), «LocalizationManager» – зчитує дані з .json файлів та завантажує їх в клас «LocalizationData» (Б.6) , «LocalizedText» – завантажує зчитаний текст до компоненту «Text» (Б.7).


Рисунок 4.4 — Створені «.json» файли

Меню, в якому відображається вибір типу питань динамічно генерується при потребі його відображення за допомогою скриптів «TaskControl» (Б.8) та «TaskToMenu» (Б.9). Для гортання списку завдань використовується компонент «Scrollbar».

Після вибору типу питань програма починає завантажувати питання та відображати список в якому реалізовано можливість перемикатися між ними. Це реалізується за допомогою 4 скриптів. «Loader» — завантажує обране завдання

(Б.10), «TaskList» — завантажує список питань (Б.11), «ListTaskControlName» — забезпечує зв’язок між активним завданням та списком завдань (Б.12), «ManageQuestion» — відповідає за роботу самого завдання зчитування відповіді та її перевірка (Б.13).

Для переключення між стартовою та основною сценами використовується скрипт «SceneManagers» (Б.14).

Відображення всіх скриптів у середовищі розробки Unity представлено на рис. 4.5.

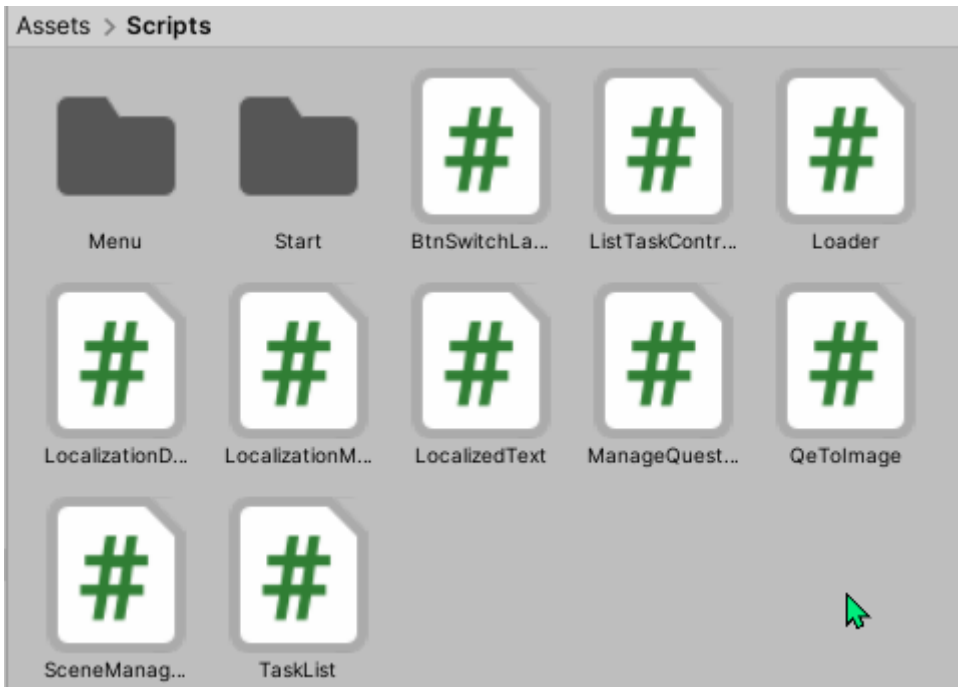


Рисунок 4.5 — Відображення скриптів у середовищі Unity

## 4.2 Опис роботи тренажеру

Після запуску програми, вітальне вікно тренажеру (рис. 4.6), де представлено тему тренажеру, автора програми, та три кнопки: «Почати» — завантажує вікно вибору типу питання, «Вихід» — закриває вікно тренажеру, «Мова» — відкриває меню зміни мови (рис. 4.7).

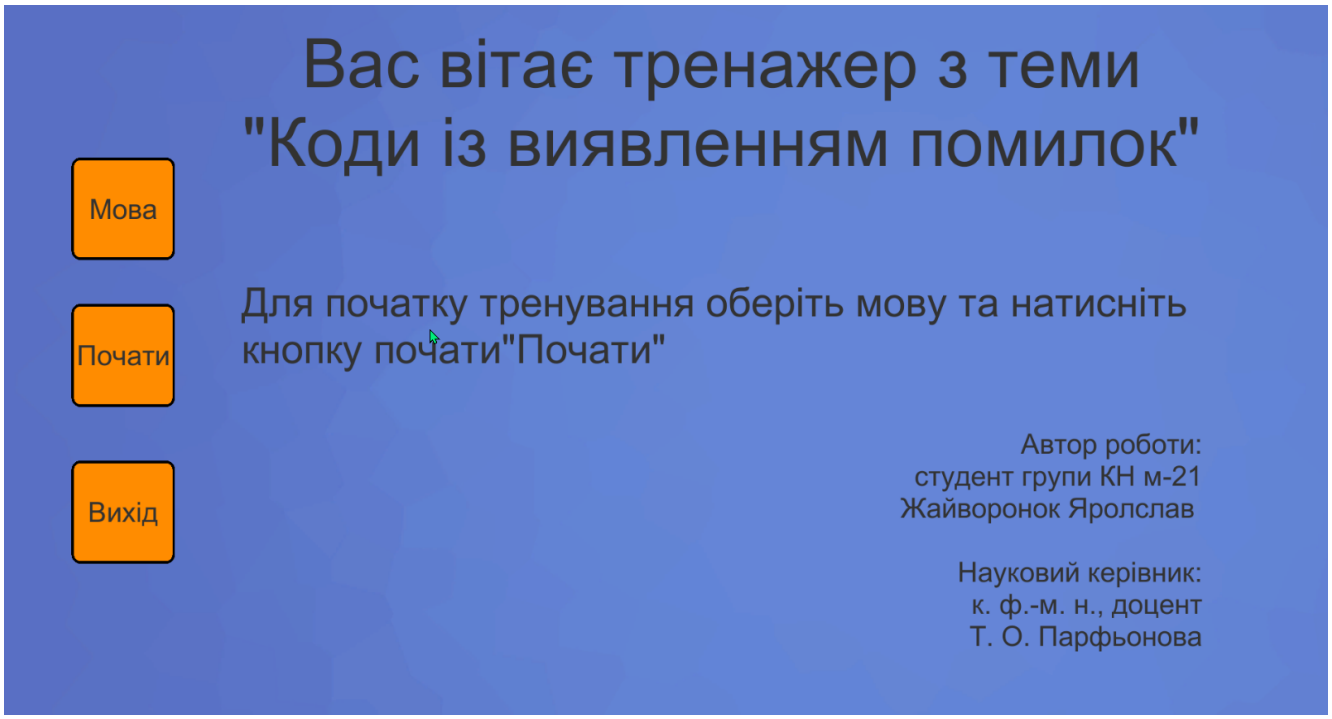


Рисунок 4.6 — Вітальне вікно тренажеру


Рисунок 4.7 — Вікно зміни мови

Коли користувач натисне кнопку «Почати» з’явиться меню (рис. 4.8) в якому представлені всі доступні теми «Кодів із виявленням помилок» та оцінка яку він отримав проходячи цю тему.


Рисунок 4.8 — Меню вибору типу питань

Після того як користувач вибере тему та натисне на обрану тему, завантажиться вікно вибору режиму тренування (практичний або теоретичний). Приклад цього вінка відображено на рисунку 4.9.


Рисунок 4.9 — Меню вибору типу питань

В залежності від вибору користувача завантажиться вікно з питаннями теми, яку обрав користувач. Під час роботи з тренажером користувачу завжди буде доступна кнопка «Довідка» (рис. 4.10) після натискання якої відкриється браузер по замовчуванню з відкритим файлом лекції, де користувач може звернутися за додатковою інформацією до лекційного матеріалу [9], який розміщено у сховищі даних Google Drive.

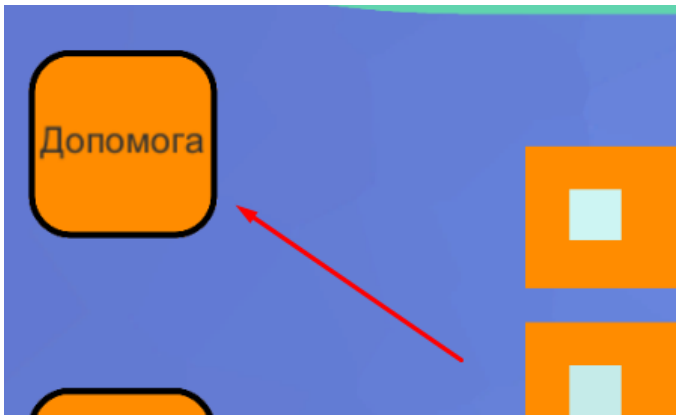


Рисунок 4.10 — Кнопка «Допомога»

Також під час тренування користувач може переглядати як наступні питання даного типу, так і ті, на які він вже відповів за допомогою списку питань, приклад якого зображено на рисунку 4.11.

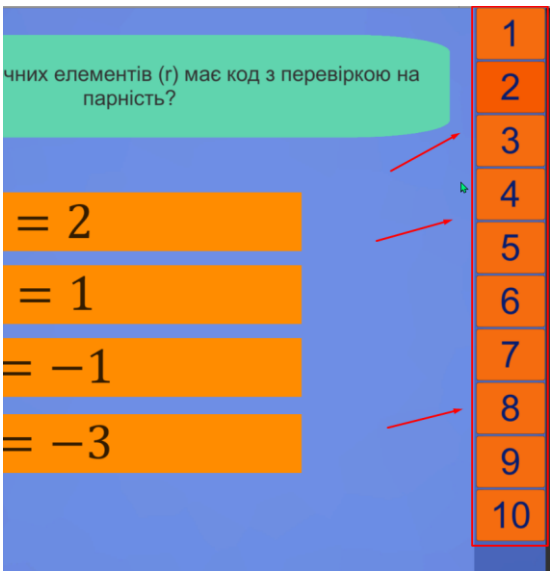


Рисунок 4.11 — Список питань

В залежності від типу питань вікно, з яким взаємодіє користувач має різний вигляд: приклад вікна з теоретичним питанням зображено на рисунку 4.12, приклад практичного питання зображено на рисунку 4.13.

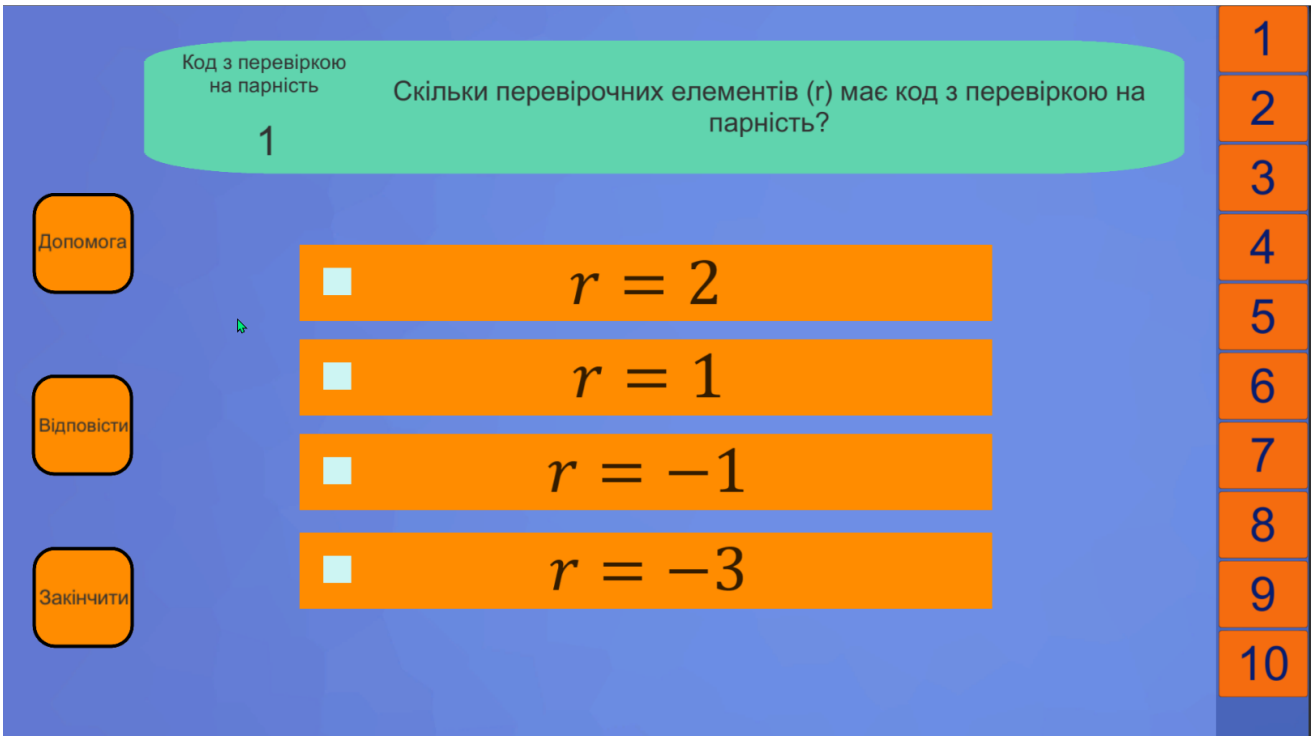


Рисунок 4.12 — Вікно з питанням тестового типу

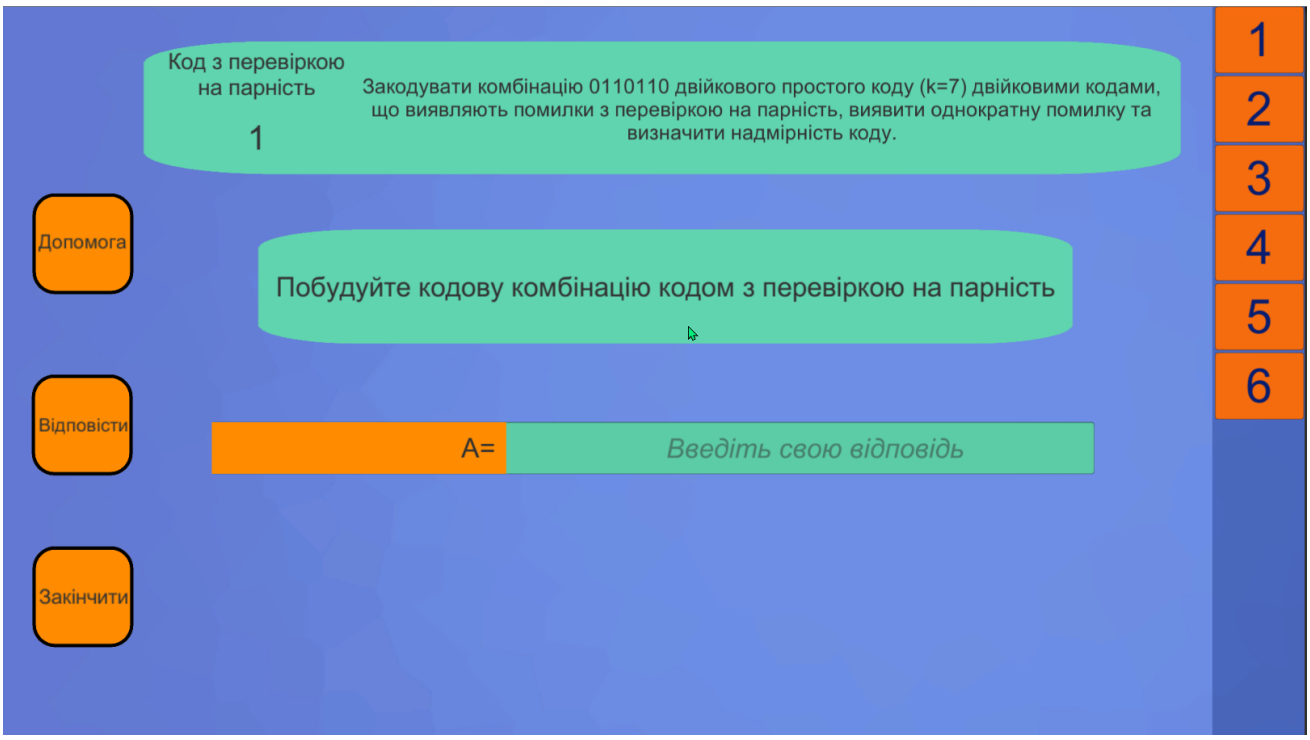


Рисунок 4.13 — Вікно з питанням практичного типу

Якщо користувач не обрав або не увів відповідь та натиснув кнопку «Відповісти» — то програма виведе повідомлення про це (рис. 4.14, 4.15).

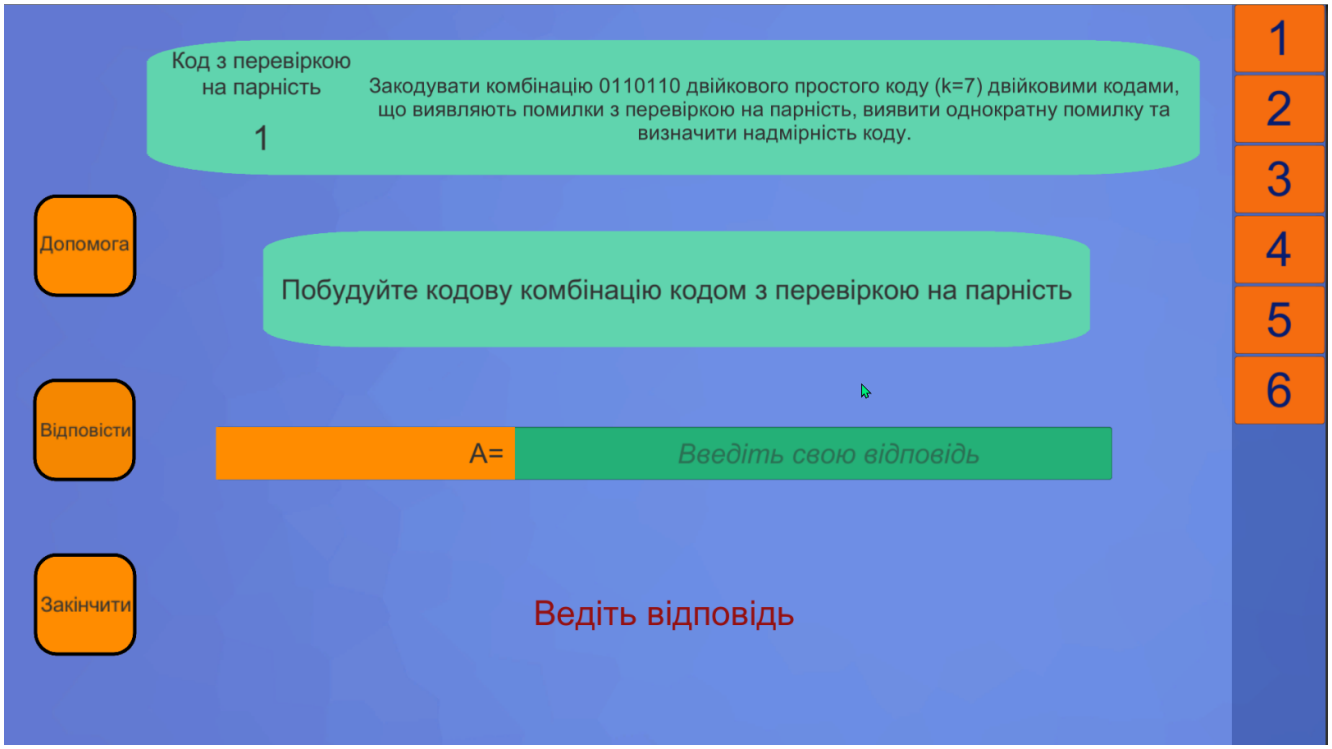


Рисунок 4.14 — Вікно з необраною відповіддю у практичному питанні


Рисунок 4.15 — Вікно з необраною відповіддю у тестовому питанні

Коли користувач обрав невірну відповідь у тестовому режимі його відповідь підсвітиться червоним фоном, а вірна зеленим (рис. 4.16). В практичному режимі

при введені невірної відповіді перший раз завантажиться питання–підказка, при введені невірної відповіді вдруге на екрані з’явиться вірна відповідь (рис. 4.17).

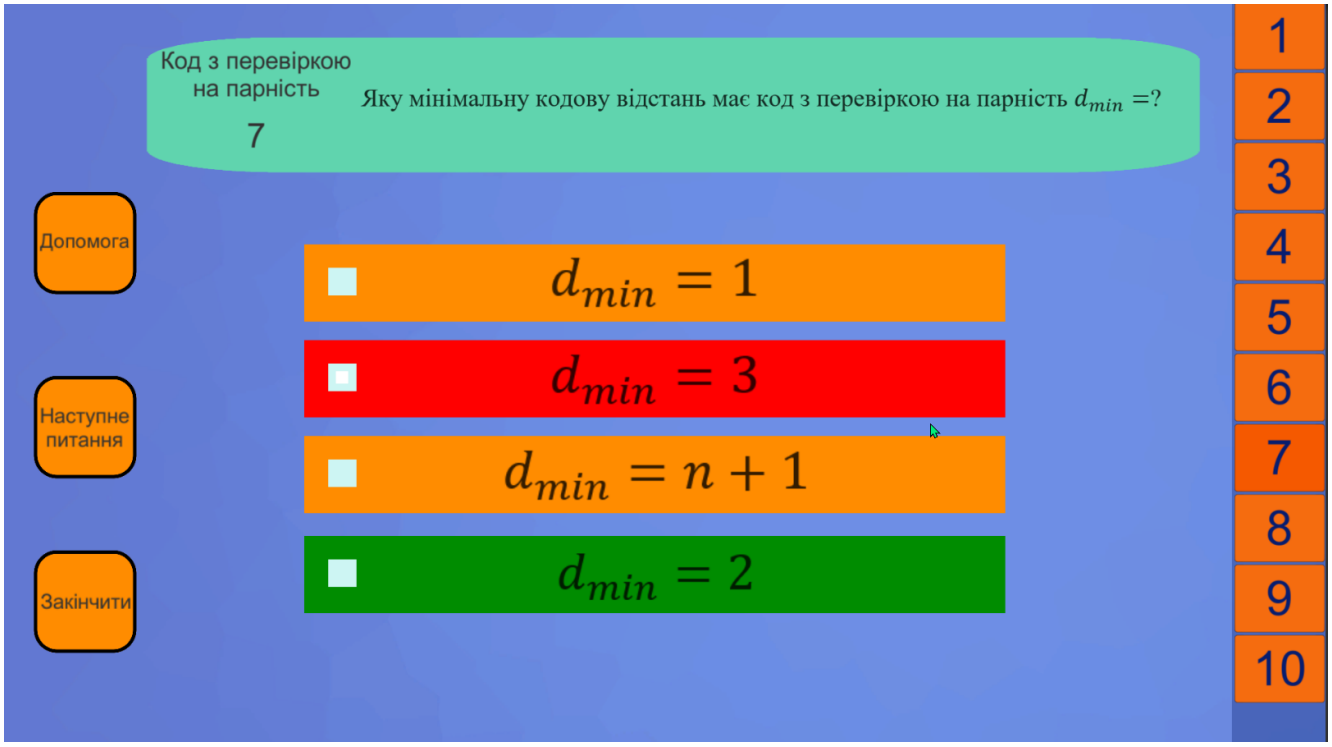


Рисунок 4.16 — Вікно тестового питання з невірно обраною відповіддю


Рисунок 4.17 — Вікно практичного типу з невірно введеною відповіддю

Коли користувач обрав вірну відповідь у тестовому режимі його відповідь підсвітиться зеленим кольором (рис. 4.18). В практичному режимі при введені вірної відповіді з’явиться повідомлення, що відповідь вірна і підсвітиться зеленим кольором (рис. 4.19).

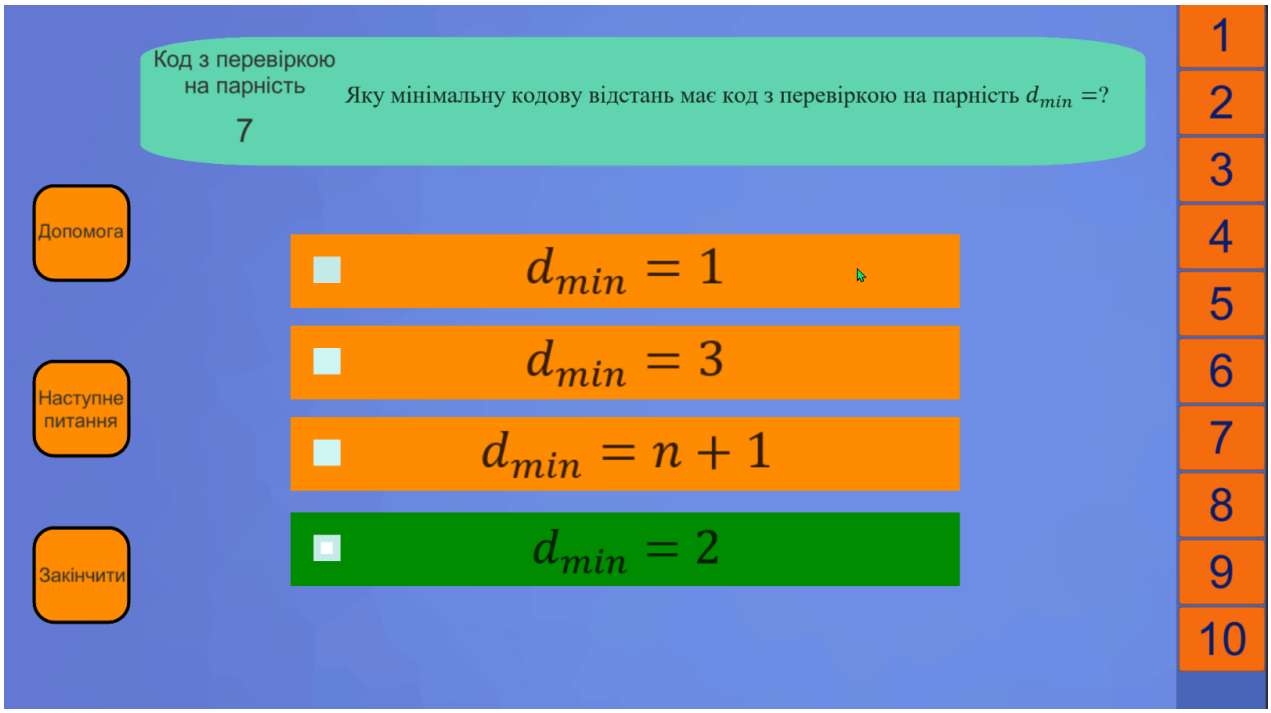


Рисунок 4.18 — Вікно тестового питання з вірно обраною відповіддю

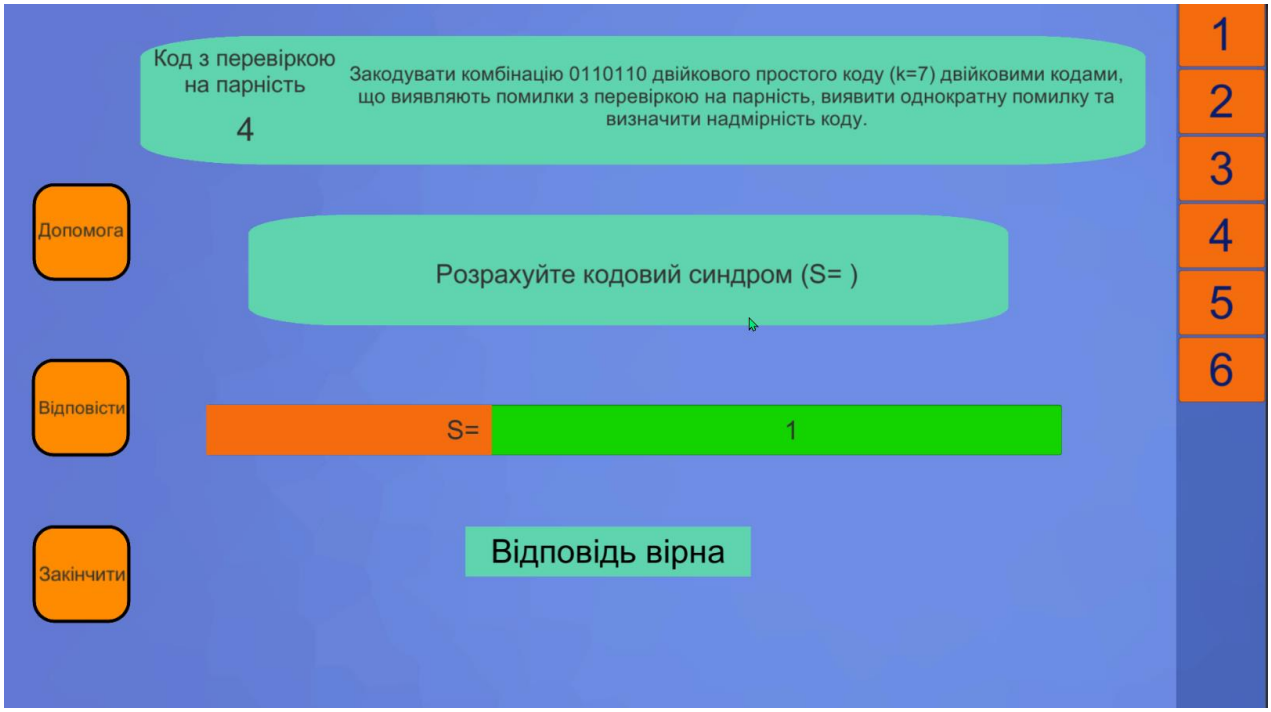


Рисунок 4.19 — Вікно практичного типу з вірно введеною відповіддю

Розрахунок балів ведеться для кожного етапу окремо - за кожну вірну відповідь користувачу зараховується один бал. Максимальна кількість балів для кожної теми дорівнює сумі кількості питань практичного та теоретичного типу. Оцінка динамічно змінюється в ході проходження тренажеру та відображається в правому нижньому кутку вікна тренажеру (рис. 4.20).

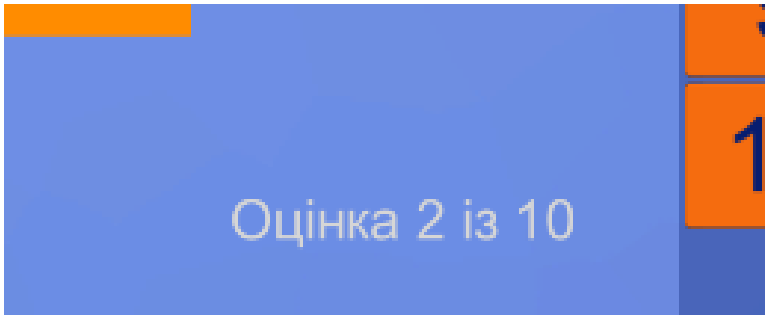


Рисунок 4.17 — Динамічне відображення оцінки

Коли користувач відповість на всі питання з’являється вікно з повідомленням про завершення тренажеру, оцінкою та трьома кнопками «Спробувати на іншому пристрої» — відкриває вікно у браузері з файлами інших версій тренажер, «Меню» — завантажує меню, «Вихід» — закриває вікно тренажеру.

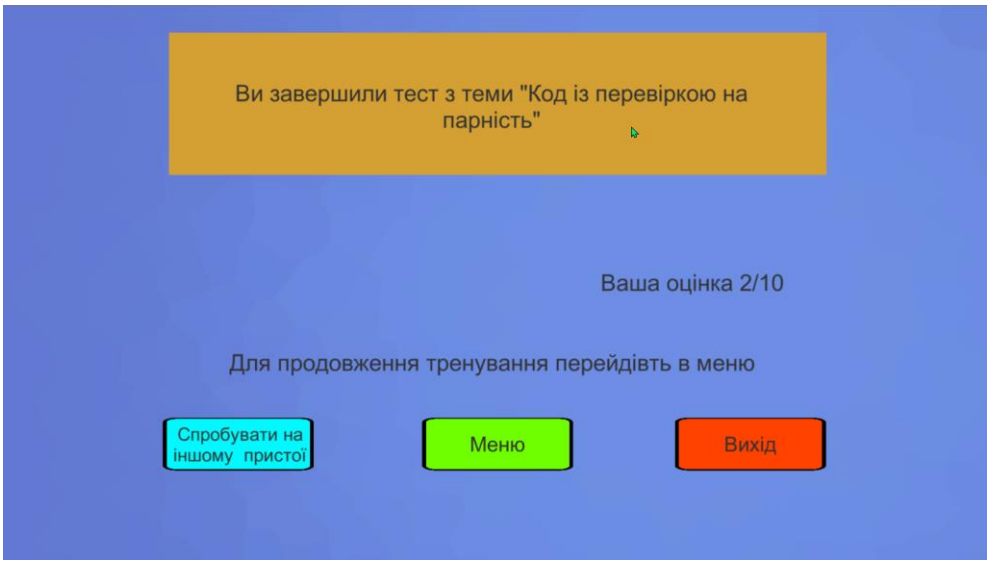


Рисунок 4.18 — Вікно після проходження одного з етапів тренування

## 4.3 Інструкція для запуску тренажеру на різних пристроях

Для запуску тренажеру на ПК потрібно:

1. Перейти за посиланням [10].

2. Завантажити архів з назвою «Error detection codes PC_x86.rar». Для завантаження файлу потрібно натиснути правою кнопкою миші та з випадаючого списку обрати пункт «Завантажити» (рис. 4.19).

3. Розпакувати архів у довільну папку.

4. Відкрити розпаковану папку та завантажити натиснувши двічі по exe – файлу з назвою «Тренажер коди із виявленням помилок» (рис. 4.20).

Або просто відкрити HTML5 версію тренажеру в браузері, яка доступна за посиланням [11] (рис. 4.21).

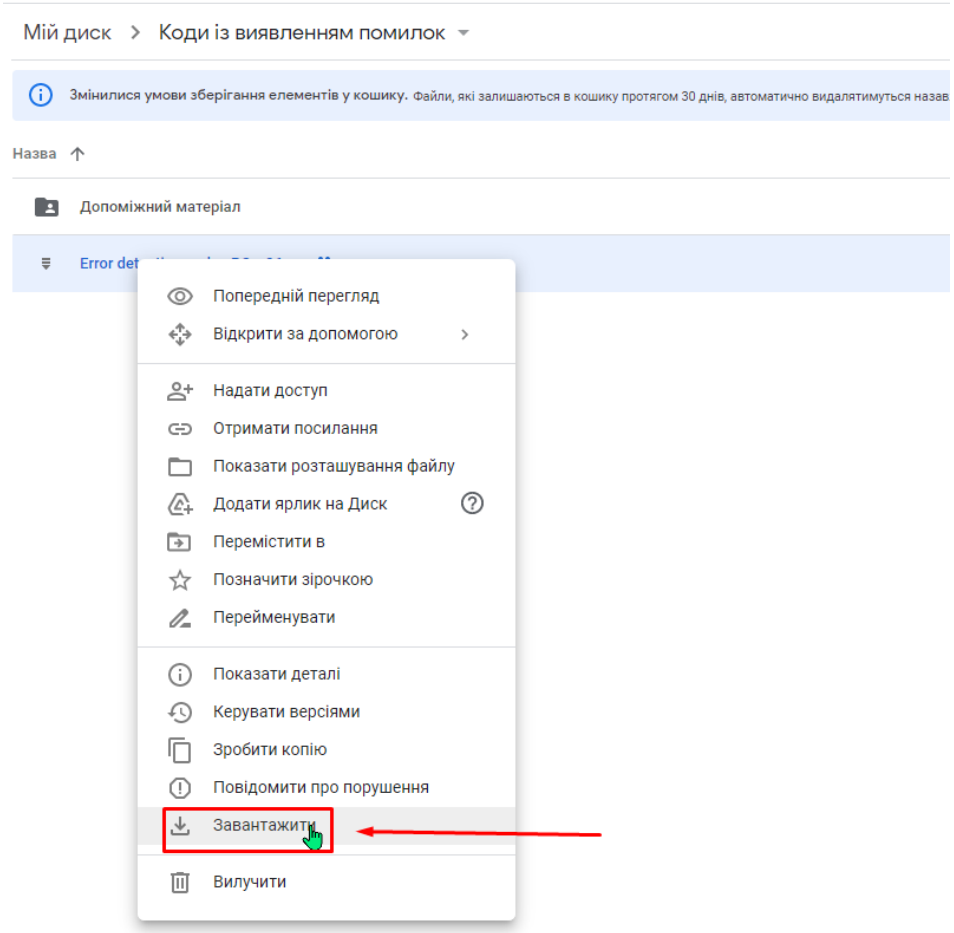


Рисунок 4.19 — Процес завантаження ПК версії тренажеру

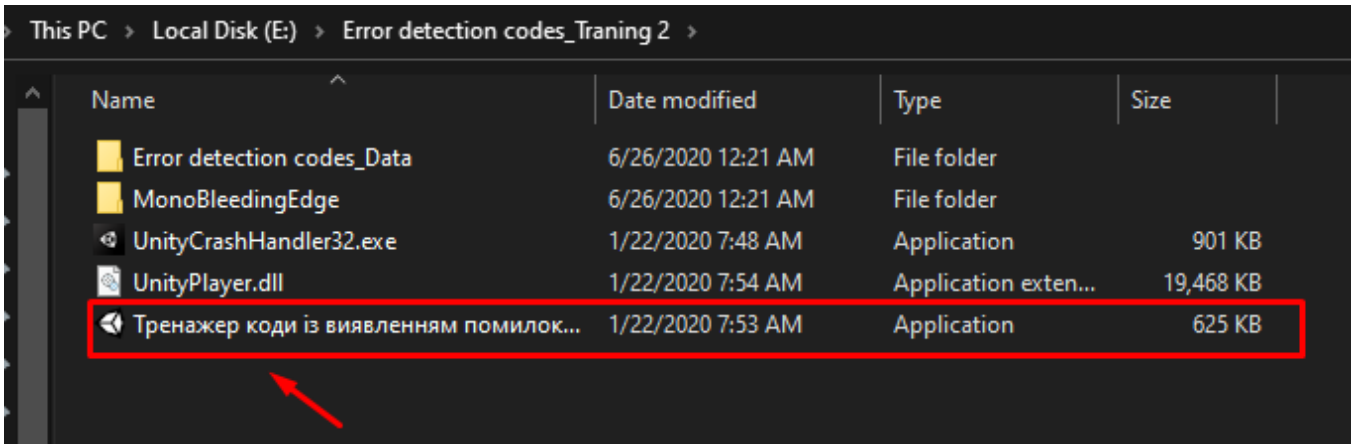


Рисунок 4.20 — Процес відкриття ПК версії тренажеру

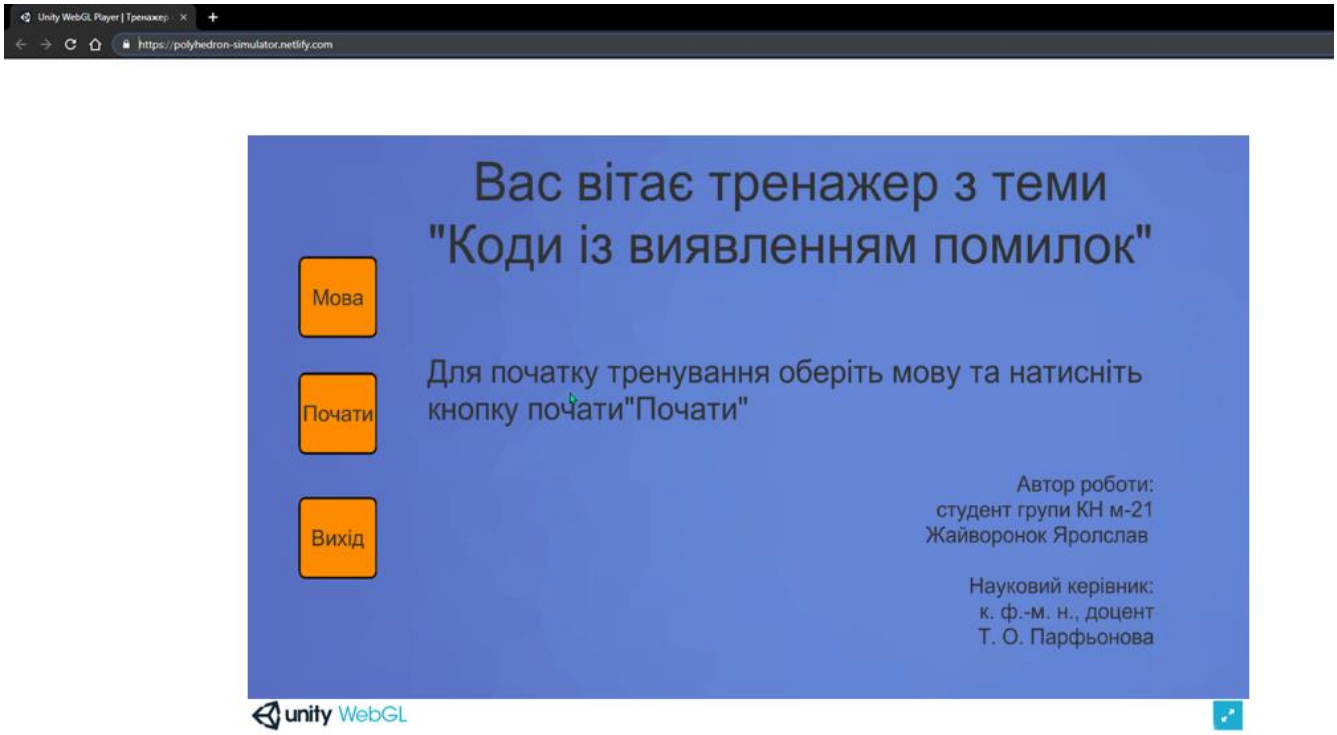


Рисунок 4.21 — Відкритий тренажер у браузері

Для запуску тренажеру на Andoroid потрібно:

1. Смартфон з операційною системою Android версії 5.0 (або вище) та увімкнену можливість встановлювати додатки з невідомих джерел.
2. Перейти за посиланням [10] на вашому смартфоні.
3. Завантажити інсталяційний APK файл «Error detection codes.apk» (рис. 4.22).

4. Після закінчення завантаження відкрити цей файл та погоджуватися з усіма подальшими питаннями програми-інсталятора.

5. Відкрити встановлений тренажер з головного екрану або з меню програм на вашому смартфоні.

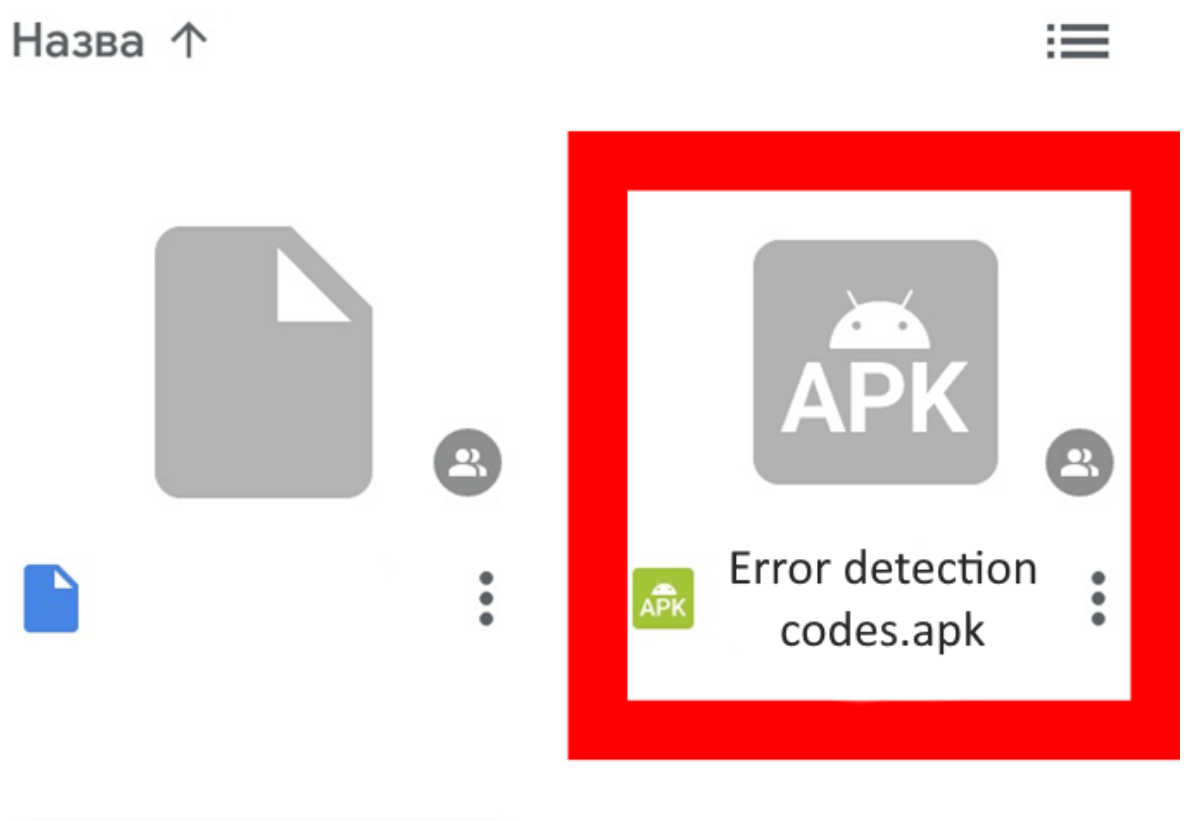


Рисунок 4.22 — Файл, який потрібно завантажити на смартфон

## 4.4 Перевірка працездатності тренажеру

Тренажер має велику кількість різних компонентів зв’язаних між собою, тому найкращим методом перевірки є почергова перевірка кожного окремого компоненту.

Перше, що було перевірено, це система зміни мови. Почергово було обрано всі доступні мови та перевірено завантаження стартового вікна, меню та кожного питання на кожній із мов. Приклад перевірки української мови представлено на рисунках 4.23 - 4.25. Приклад перевірки російської мови представлено на рисунках

4.26 - 4.29. Приклад перевірки англійської мови представлено на рисунках 4.30 - 4.32. В результаті перевірки помилок в роботі цього компоненту не виявлено.

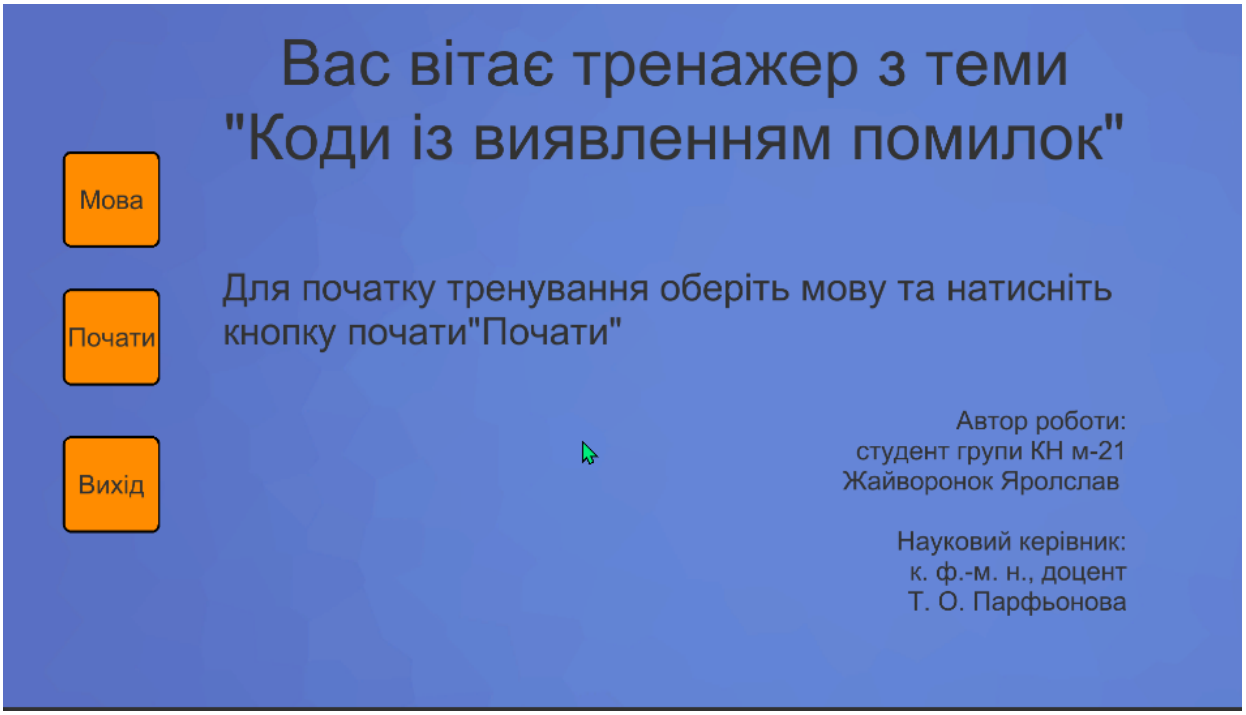


Рисунок 4.23 — Стартове вікно українською мовою


Рисунок 4.24 — Меню українською мовою

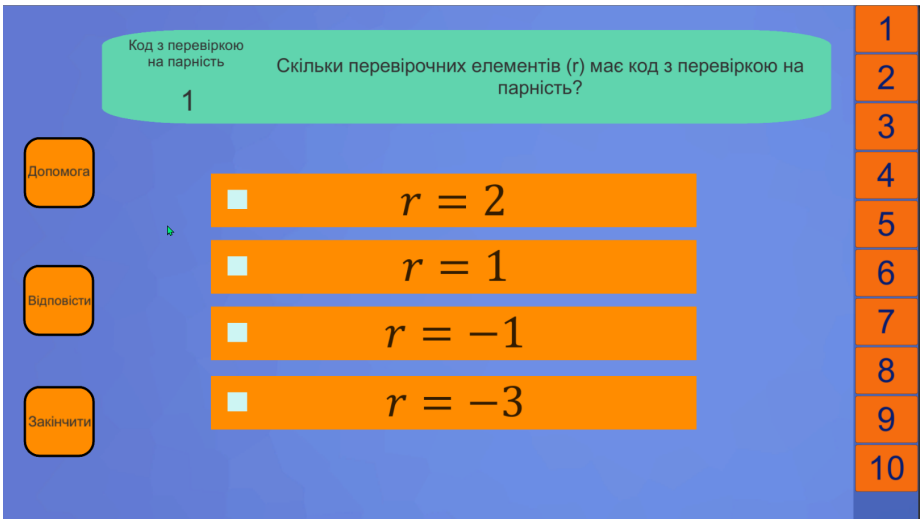


Рисунок 4.25 — Приклад одного із питань українською мовою

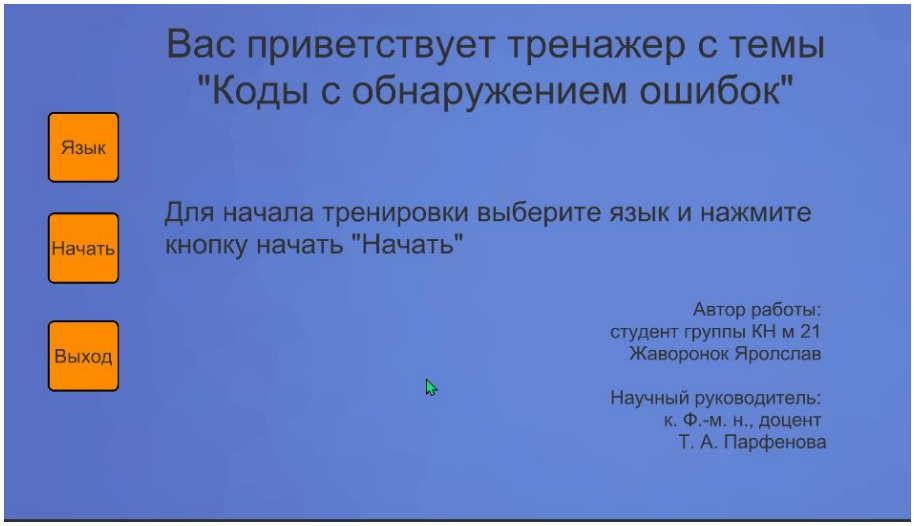


Рисунок 4.26 — Стартове вікно російською мовою


Рисунок 4.27 — Меню російською мовою

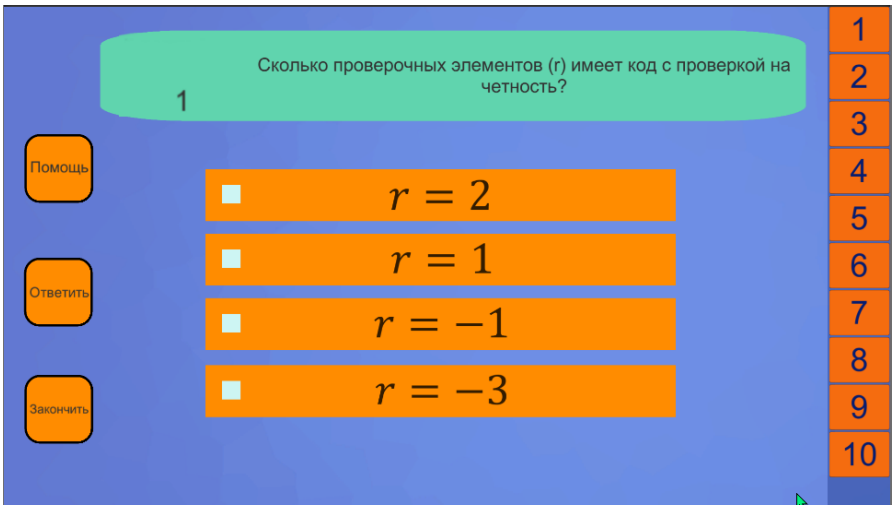


Рисунок 4.28 — Приклад одного із питань російською мовою

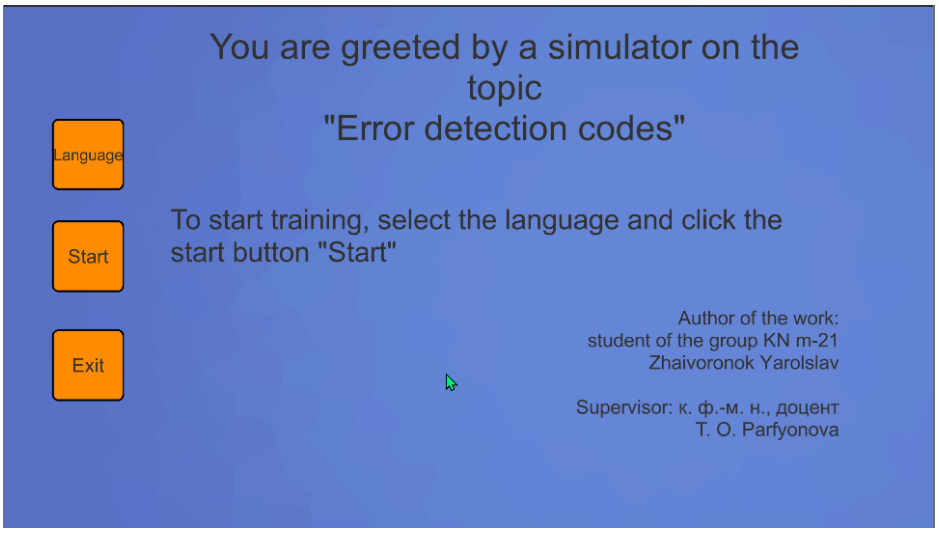


Рисунок 4.29 — Стартове вікно англійською мовою


Рисунок 4.30 — Меню англійською мовою

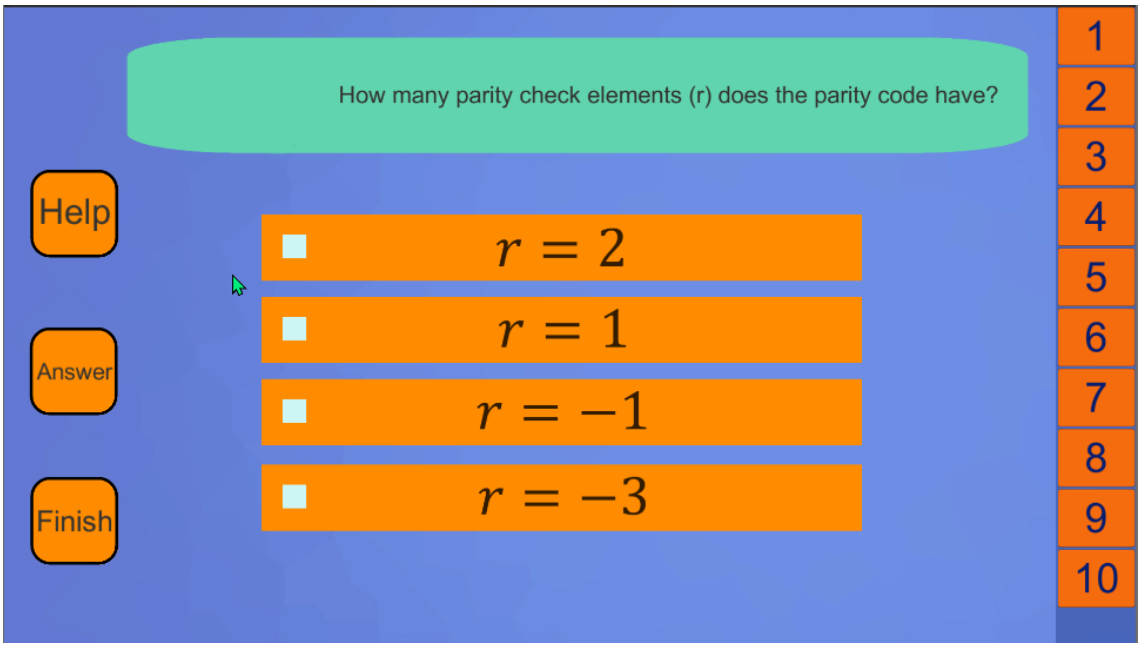


Рисунок 4.31 — Приклад одного із питань англійською мовою

Потім було перевірено роботу меню та списку питань. Почергово було запущено всі типи питань та перевірено можливість перемикатися між самими питаннями. Перевірено кількість питань у файлі з питаннями та кількість, яка відображається у списку питань (рис 4.32 – 4.33). В результаті перевірки кількість питань у файлі та кількість питань завантажених в список питань завжди була однаковою.

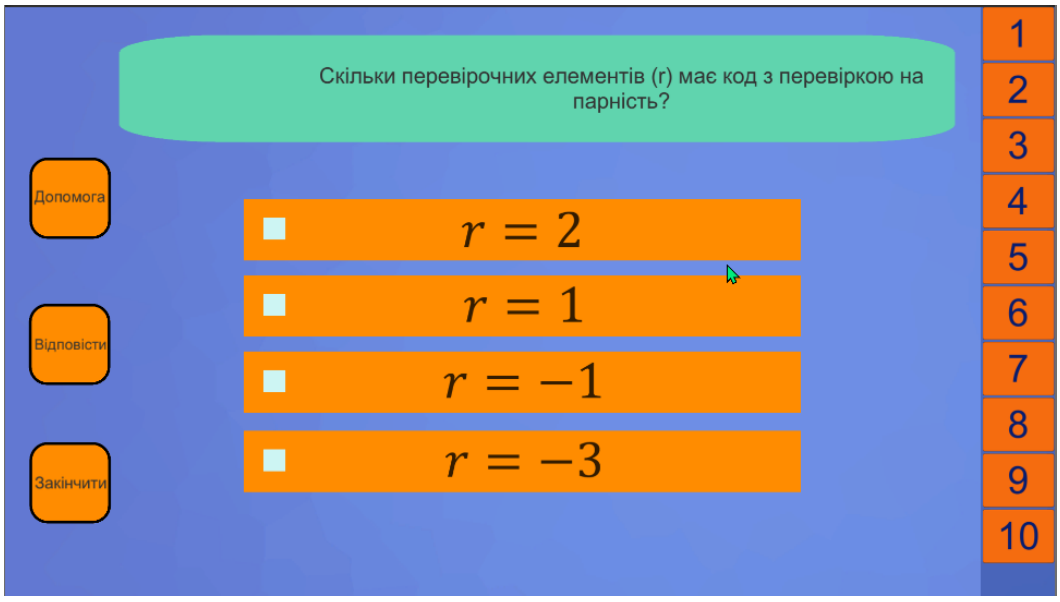


Рисунок 4.32 — Приклад списку питань який містить десять запитань

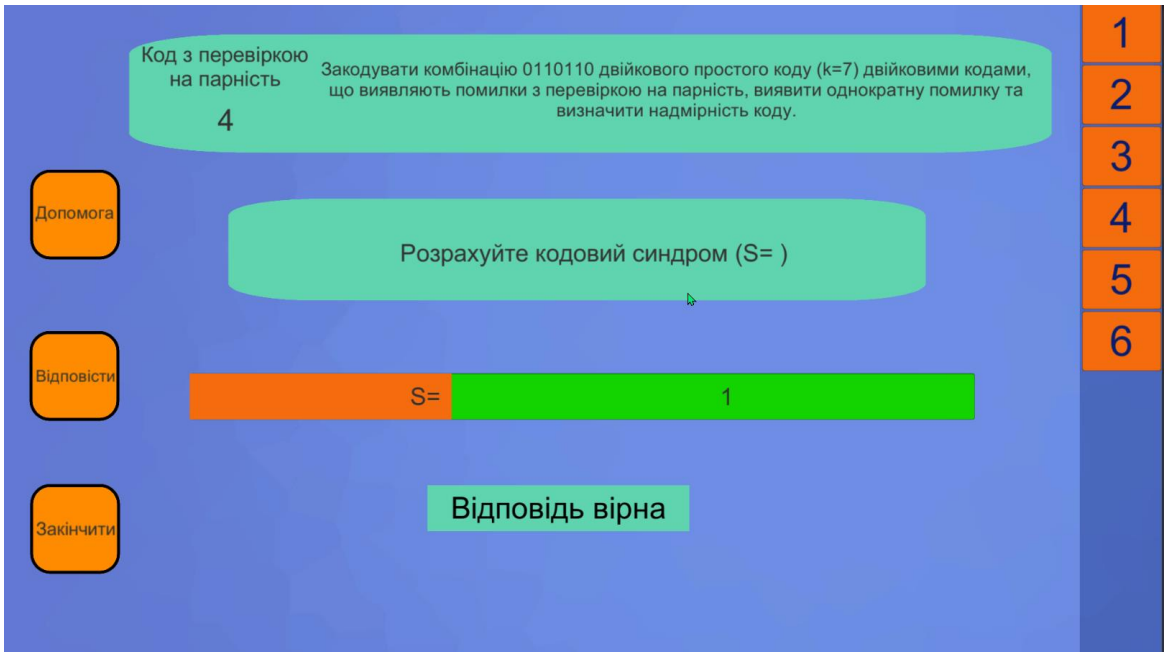


Рисунок 4.33 — Приклад списку питань який містить шість запитань

Важливим компонентом тренажеру є обробник відповідей, тому також було проведено його перевірку. Для всіх питань було перебрано всі можливі відповіді, приклад цього наведено на рисунках 4.34 та 4.35 для тестового питання, 4.36 та 4.37 для практичного питання. В результаті перевірки збоїв у роботі не виявлено, відображалися саме ті відповіді, які зазначені у файлі.

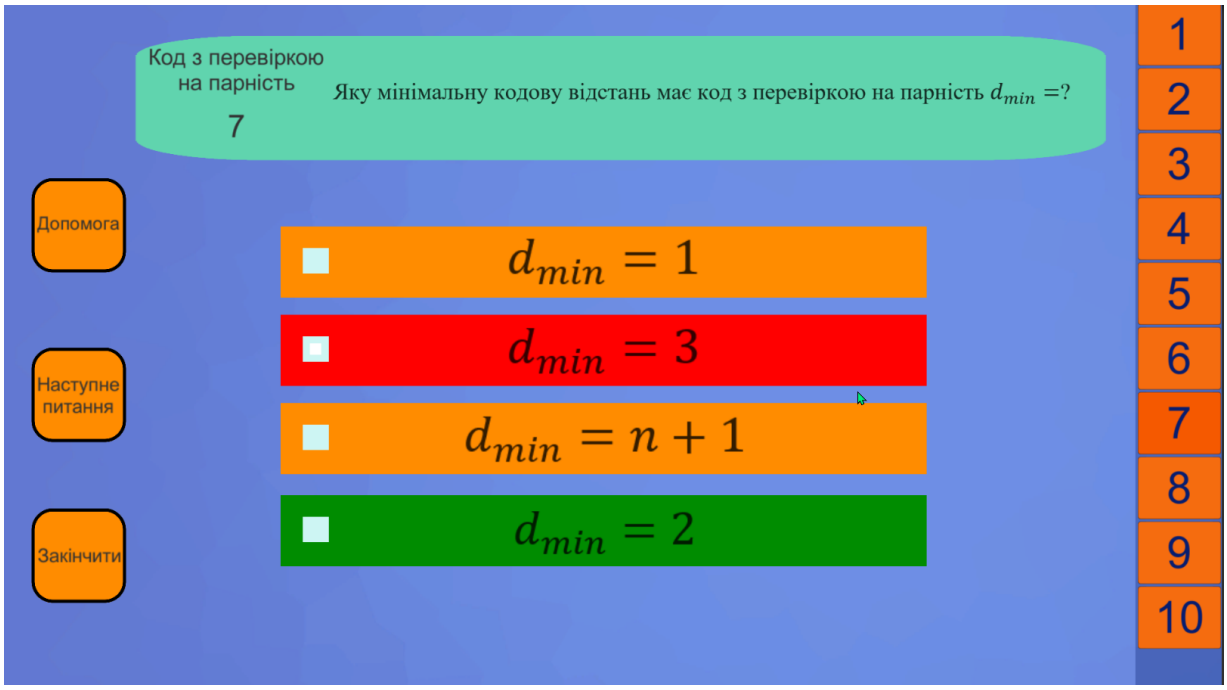


Рисунок 4.34 — Приклад тестового питання з невірно обраною відповіддю

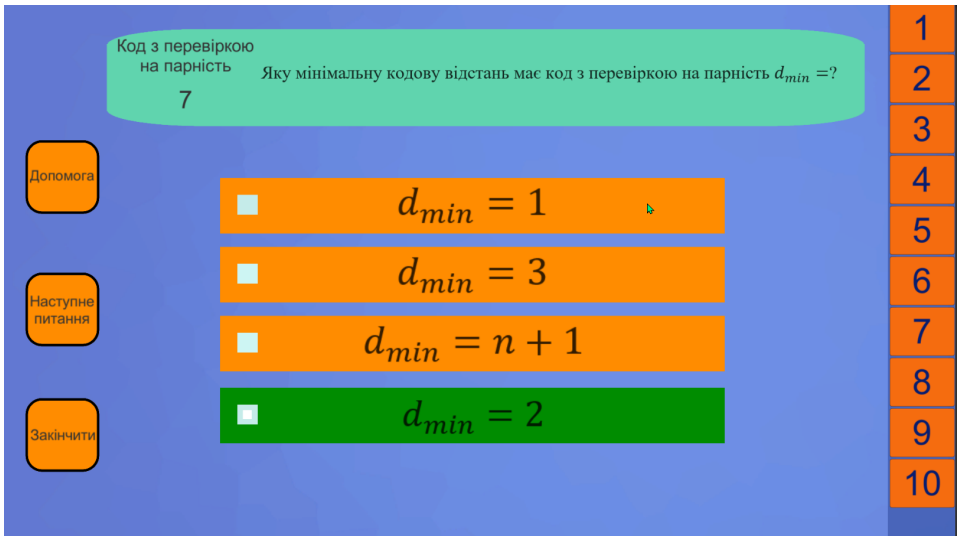


Рисунок 4.35 — Приклад тестового питання з вірно обраною відповіддю


Рисунок 4.36 — Приклад практичного питання з невірно введеною відповіддю


Рисунок 4.37 — Приклад практичного питання з вірно введеною відповіддю

Перевірено підтримку різних екранів, програму було запущено на різних пристроях з різним співвідношенням сторін екрану. Перевірено співвідношення 21:9 (рис 4.38), 16:9 (рис 4.39) та 4:3 (рис 4.40).

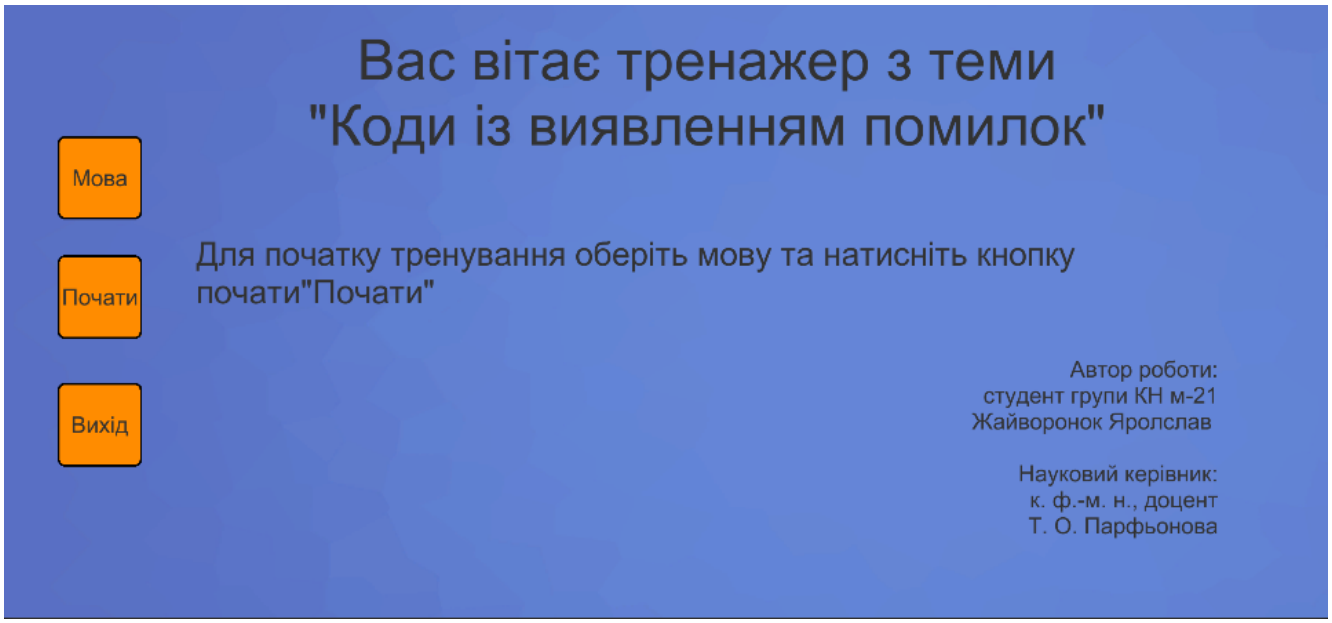


Рисунок 4.38 — Вікно програми на екрані з співвідношенням сторін 21:9

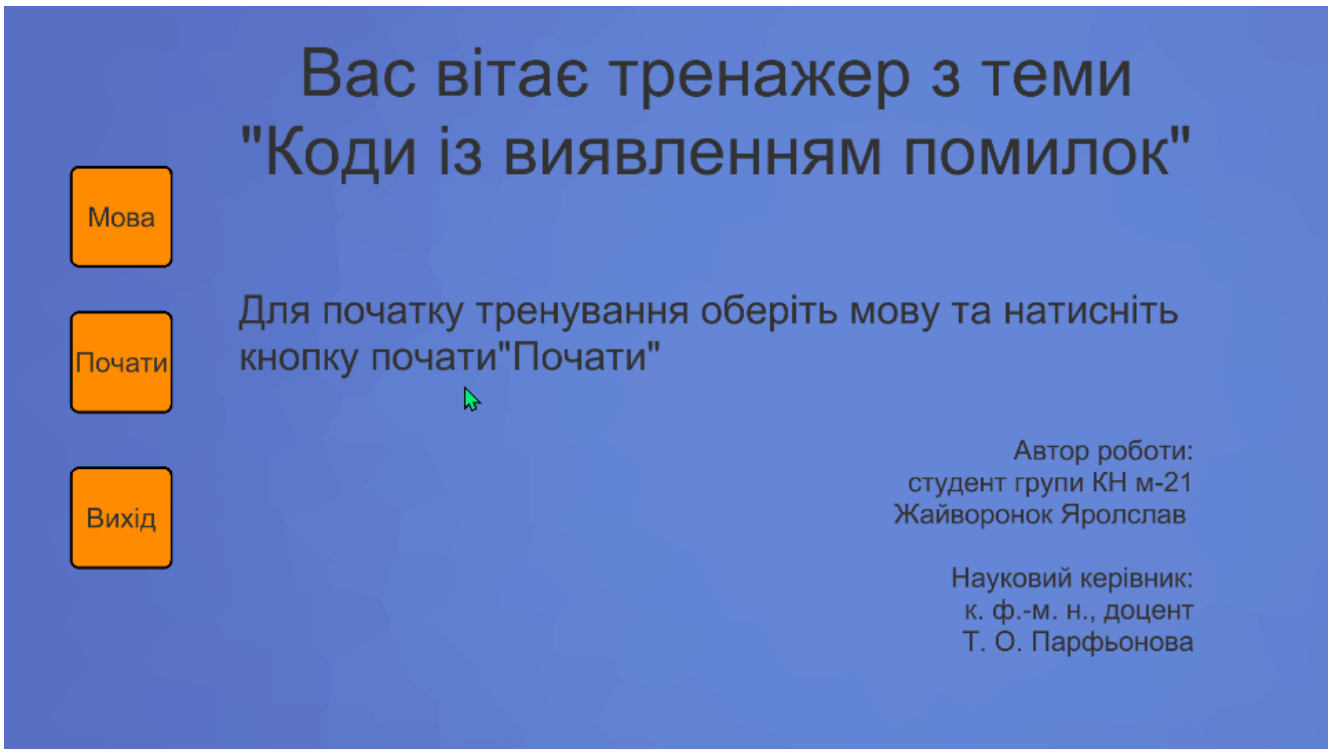


Рисунок 4.39 — Вікно програми на екрані з співвідношенням сторін 16:9

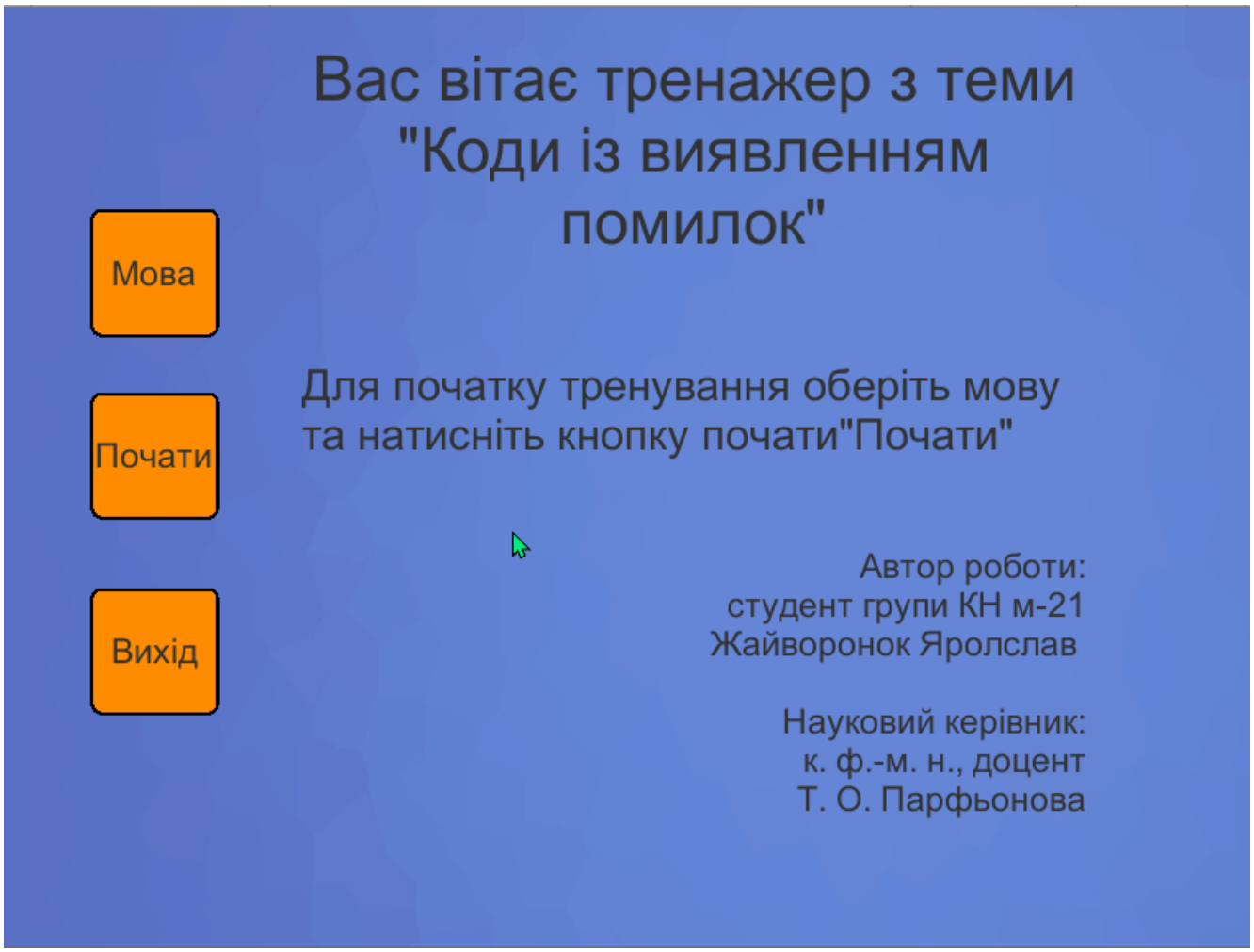


Рисунок 4.40 — Вікно програми на екрані з співвідношенням сторін 4:3

Також перевірено роботу системи розрахунку оцінок для кожного типу питань, працездатність кожної кнопки та поля. Перевірено загальну працездатність на всіх зазначених платформах.

# ВИСНОВКИ

З метою полегшення та пришвидшення здобуття знань студентом створено алгоритм та програму з теми «Розробка програмного забезпечення для тренажера дистанційного навчального курсу «Теорія інформації та кодування» з теми «Коди із виявленням помилок».

В ході роботи над реалізацією алгоритму та програмою-тренажером:

- розроблений алгоритм, за яким студенти будуть вивчати коди із виявленням помилок містить в собі інформацію та завдання по всім основним кодам, що виявляють помилки, а саме двійкові та недвійкові коди та їх різновидності.
- розроблено блок-схему алгоритму;
- створено простий та інтуїтивно зрозумілий інтерфейс;
- реалізовано підтримку різних розширень екрану, а саме 21:9, 16:9, 4:3.
- створено скрипти перевірки відповідей користувача та виведення правильної відповіді на екран;
- розроблено систему динамічного розрахунку кількості набраних балів;
- розроблено два типи питань (теоретичний та практичний);
- створено список завдань, з допомогою якого студент зможе перемикатися між запитаннями;
- реалізовано кнопку довідки, що відкриває сторінку з даними лекції;
- виконано обробку всіх виключних ситуацій та можливих збоїв у роботі програми;
- скомпільовано програму на три обрані операційні системи, а саме Windows, Android та HTML5 для забезпечення максимального охоплення користувачів;
- тренажер завантажено у систему Moodle ПУЕТ;
- проведено інформаційний огляд та аналіз трьох робіт виконаних іншими студентами, в результаті чого виявлено їх переваги та недоліки;

- сформовано інструкцію з описом процесу завантаження та встановлення тренажеру.

Розробляючи тренажер поглиблено знання в кодах із виявленням помилок та мови програмування C#. Також вдосконалено навички розробки застосунків середовищі Unity.

За посиланням [10] у вільному доступі знаходиться Android та Windows версії застосунку. HTML5 версія тренажера доступна в інтернеті за посиланням [11]. Тренажер додано до системи Moodle ПУЕТ.

# ЛІТЕРАТУРА

1. Программирование на C# в Unity для начинающих [Електронний ресурс] // Unity Technologie. – 2020. – Режим доступу до ресурсу: https://unity3d.com/ru/learning-c-sharp-in-unity-for-beginners

2. Жураковський Ю. П. Теорія інформації та кодування / Ю. П. Жураковський, В. П. Полторак., 2001. – 134 - 157 с.

3. Бондаренко І. М. Коди та кодування / І. М. Бондаренко, Ю. П. Жураковський, А. П. Глушко. – Харків: XI ВПС, 2003. – 58 - 65 с. – (Харківський інститут ВПС ім. І. Кожедуба).

4. Обнаружение и исправление ошибок [Електронний ресурс] // Wikipedia. – 2020. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BD%D0%B0%D1%80%D1%83%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B8_%D0%B8%D1%81%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BE%D1%88%D0%B8%D0%B1%D0%BE%D0%BA.

5. Марченко Д. А. Алгоритмізація та програмна реалізація тренажера з теми «Дослідження на збіжність числових рядів» дистанційного навчального курсу «Математичний аналіз» / Д. А. Марченко // Інформатика та системні науки (ІСН-2015): матеріали VI Всеукраїнської науково-практичної конференції за міжнародною участю, (м. Полтава, 19–21 берез. 2015 р.). – Полтава: ПУЕТ, 2015. – Режим доступу до ресурсу: http://dspace.puet.edu.ua/handle/123456789/2500 .

6. Белінська В. В. Програмна реалізація тренажера для методу потенціалів лінійної задачі про оптимальний потік з дисципліни «Методи оптимізації та дослідження операцій» / В. В. Белінська // Комп’ютерні науки і прикладна математика (КНІПМ-2019): матеріали науково-практичного семінару, випуск 3, (м. Полтава січень - червень 2019 р.) – Полтава: ПУЕТ, 2019. – Режим доступу до ресурсу: http://dspace.puet.edu.ua/handle/123456789/8278

7. Самовик С.М. Розробка алгоритму та програмного забезпечення тренажера з теми "Угорський метод в задачі про призначення" дистанційного

навчального курсу "Методи оптимізації та дослідження операцій" / С.М. Самовик // Інформатика та системні науки (ІСН-2014) : матеріали V Всеукр-.наук.-практ. конф., (м. Полтава, 13–15 березня 2014 р.). – Полтава: ПУЕТ, 2014. – С. 273-274. – Режим доступу до ресурсу: http://dspace.puet.edu.ua/handle/123456789/2840

8. Scripting API [Електронний ресурс] // Unity Technologies. – 2020. – Режим доступу до ресурсу: https://docs.unity3d.com/2019.1/Documentation/ScriptReference/

9. Допоміжний матеріал [Електронний ресурс]. – 2020. – Режим доступу до ресурсу:https://drive.google.com/drive/folders/1dyx4iw_D5dIpOZ5Y4w1BPJ9z8B0TriOs?usp=sharing

10. Android та Windows версії тренажеру [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://drive.google.com/drive/folders/1q3FmH-h9jAe-XELDpX88aekor0APItNz?usp=sharing

11. HTML версія тренажеру [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://Error-detection-codes.netlify.com/

# ДОДАТОК А
# БЛОК-СХЕМА

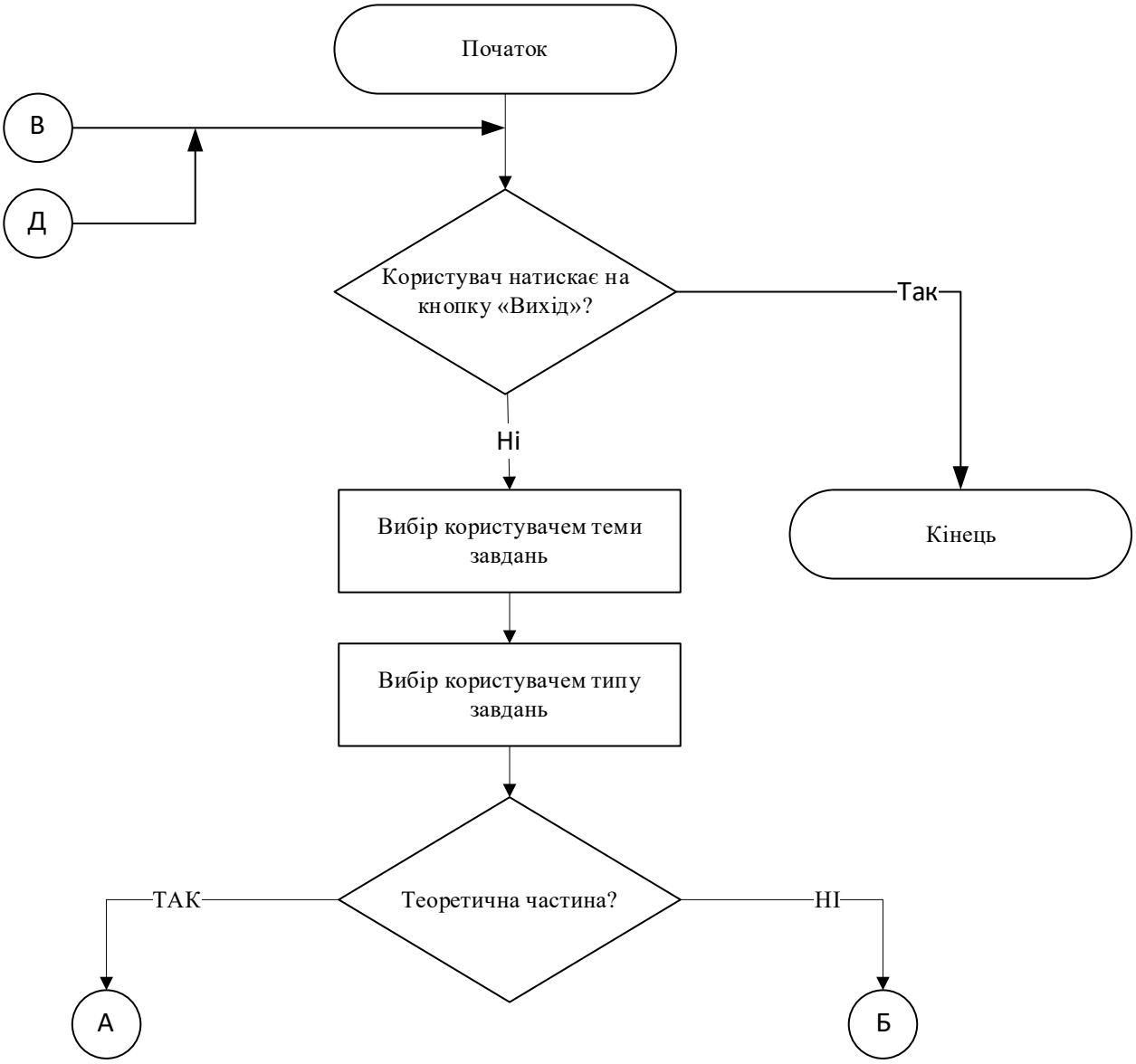


Рисунок А.1 — Основна частина блок-схеми алгоритму навчального тренажера

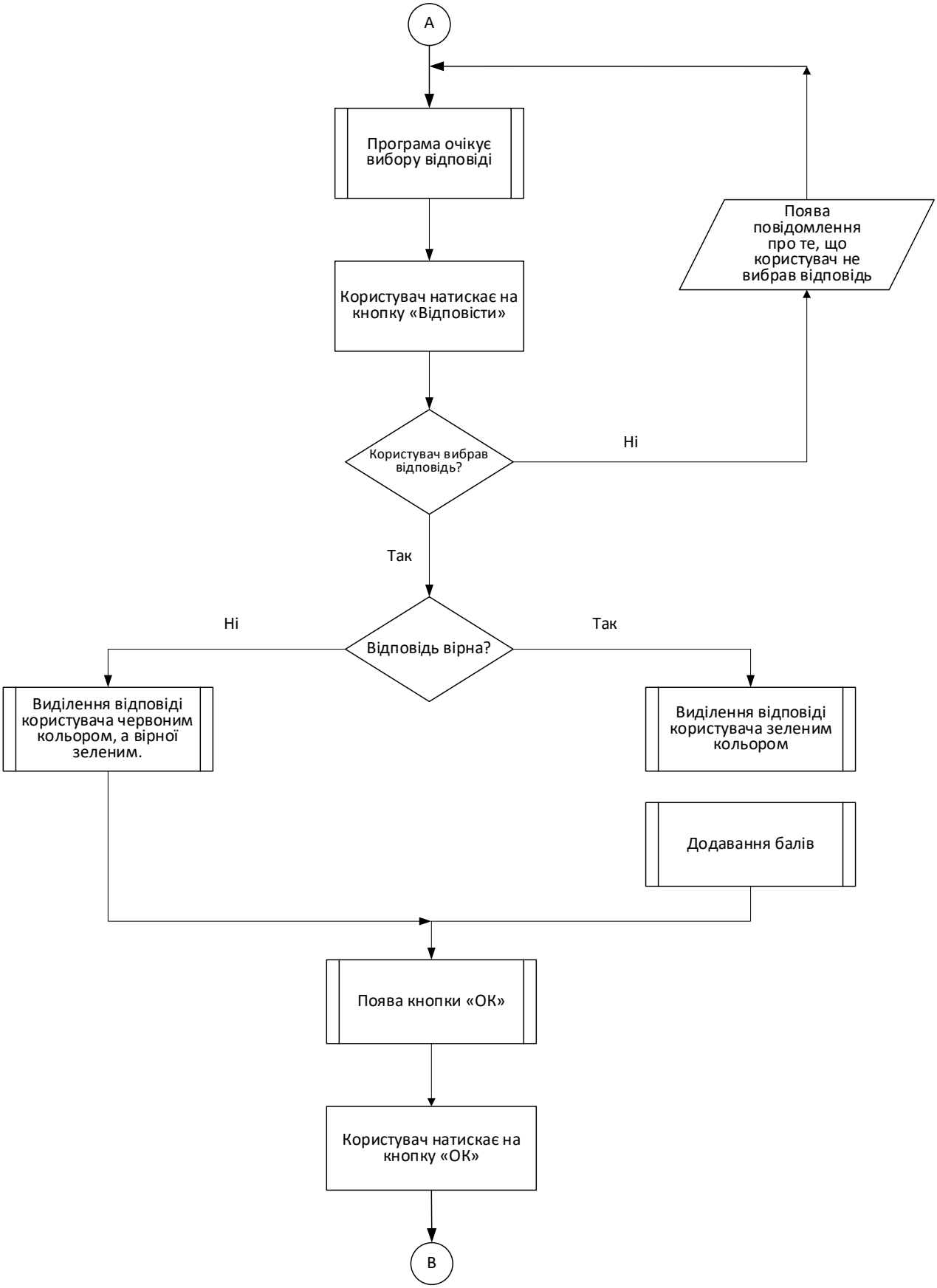


Рисунок А.2 — Алгоритм обробки питання теоретичної частини навчального тренажера

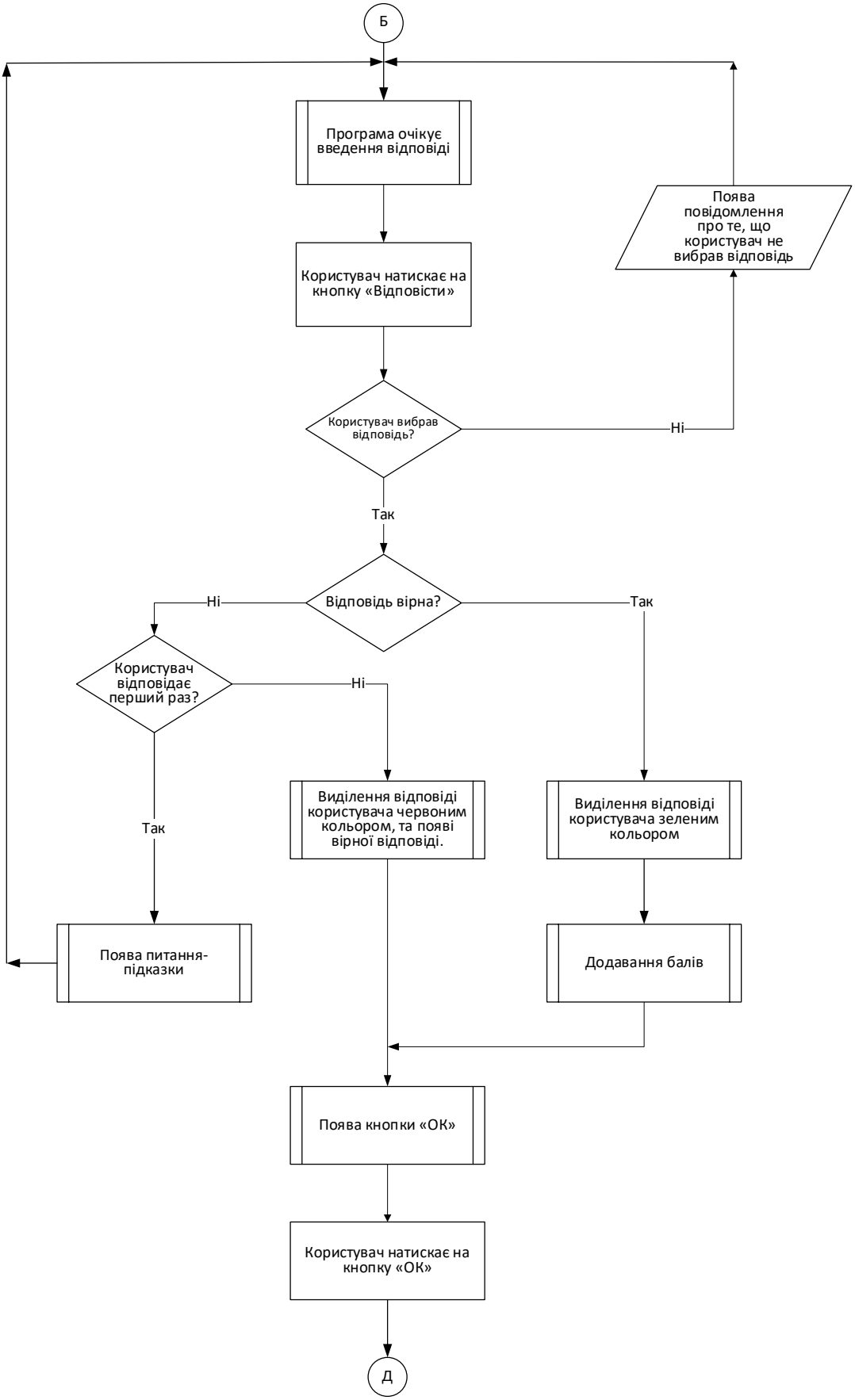


Рисунок А.3 — Алгоритм обробки питання практичної частини навчального тренажера

# ДОДАТОК Б
# КОД СКРИПТІВ ПРОГРАМИ

## Б.1 Вміст файлу «ua_UA.json»

```
{
  "items": [
    {
      "key": "null",
      "value": "Пусте поле"
    },
    {
      "key": "ButLangOk",
      "value": "Обрати"
    },
    {
      "key": "ButNextQe",
      "value": "Наступне питання"
    },
    {
      "key": "ButEXIT",
      "value": "Вихід"
    },
    {
      "key": "ButHELP",
      "value": "Допомога"
    },
    {
      "key": "ButOK",
      "value": "Відповісти"
    },
    {
      "key": "ButStart",
      "value": "Почати"
    },
    {
      "key": "ButLang",
      "value": "Мова"
    },
    {
      "key": "ButStop",
      "value": "Закінчити"
    },
    {
      "key": "TextTopic",
      "value": "Вас вітає тренажер з теми\n\"Коди із виявленням помилок\""
    },
    {
      "key": "TextTooltip",
      "value": "Для початку тренування оберіть мову та натисніть кнопку почати\"Почати\""
    },
    {
      "key": "TextForAutor",
      "value": "Автор роботи:\n студент групи КН м-21\n Жайворонок Яролслав \n \n Науковий
керівник:\n к. ф.-м. н., доцент\nТ. О. Парфьонова"
    },
    {
      "key": "EndWindowTitle",
      "value": "Вітаємо ви завершили проходження теми: "
    },
    {
      "key": "EndWindowBals",
      "value": "Ваша оцінка: "
    }
  ],
  "question": [
    {
```

```
      "qeTypeNum": "0",
      "regime": "test",
      "qeTypeName": "Код з перевіркою на парність",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [
        {
          "qeNumQue": "0",
          "question": "Скільки перевірних елементів (r) має код з перевіркою на парність?",
          "correctANS": "2",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "1",
          "regime": "test",
          "question": "Скільки інформаційних елементів (k) має код з перевіркою на парність?",
          "correctANS": "1",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "2",
          "regime": "test",
          "question": "Як позначається код з перевіркою на парність?",
          "correctANS": "3",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "3",
```

```
          "regime": "test",
          "question": "Довжина кодової комбінації (n) обраховується за формулою?",
          "correctANS": "3",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "4",
          "regime": "test",
          "question": "Який вигляд має формула для створення перевірного елементу в коді з
перевіркою на парність?",
          "correctANS": "1",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "5",
          "regime": "test",
          "question": "Як утворюється кодова комбінація в коді з перевіркою на парність?",
          "correctANS": "1",
          "ansver": [
            {
              "ansver": "Кодова комбінація утворюється доповненням комбінації k – елементного
первинного коду одним елементом таким чином, щоб кількість одиниць у новому n – розрядному ( n = k
+1) коді була парною."
            },
            {
              "ansver": "Кожна комбінація має непарну кількість одиниць, тобто додатковий перевірний
елемент формують, виходячи з кількості одиниць у початковій кодовій комбінації; при парній кількості
перевірний елемент дорівнює одиниці, а при непарній – нулю."
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "6",
          "regime": "test",
          "question": "img",
          "correctANS": "4",
          "ansver": [
            {
              "ansver": "img"
```

```
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    }
  ]
},
{
  "qeNumQue": "7",
  "regime": "test",
  "question": "img",
  "correctANS": "3",
  "ansver": [
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    }
  ]
},
{
  "qeNumQue": "8",
  "regime": "test",
  "question": "Відсутність помилки в комбінації вважається коли? ",
  "correctANS": "1",
  "ansver": [
    {
      "ansver": "S=0"
    },
    {
      "ansver": "S=1"
    },
    {
      "ansver": "S=-1"
    },
    {
      "ansver": "S=2"
    }
  ]
},
{
  "qeNumQue": "9",
  "regime": "test",
  "question": "Яким виразом визначається надмірність коду?",
  "correctANS": "3",
  "ansver": [
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
```

```
            "ansver": "img"
          }
        ]
      }
    ],
    "questionDetailsPr": [
      {
        "qeNumQue": "0",
        "regime": "pr",
        "question": "Побудуйте кодову комбінацію кодом з перевіркою на парність",
        "correctANSStr": "01101100",
        "hint": [
          {
            "hintQeNumType": "1",
            "hintNumQe": "5"
          },
          {
            "hintQeNumType": "1",
            "hintNumQe": "6"
          }
        ]
      },
      {
        "qeNumQue": "1",
        "regime": "pr",
        "question": "Розрахуйте довжину кодової комбінації (n=)",
        "correctANSStr": "8",
        "hint": [
          {
            "hintQeNumType": "1",
            "hintNumQe": "4"
          }
        ]
      },
      {
        "qeNumQue": "2",
        "regime": "pr",
        "question": "img",
        "correctANSStr": "01001100",
        "hint": [
          {
            "hintQeNumType": "0",
            "hintNumQe": "0"
          }
        ]
      },
      {
        "qeNumQue": "3",
        "regime": "pr",
        "question": "Розрахуйте кодовий синдром (S=)",
        "correctANSStr": "1",
        "hint": [
          {
            "hintQeNumType": "1",
            "hintNumQe": "8"
          }
        ]
      },
      {
        "qeNumQue": "4",
        "regime": "test",
        "question": "Використавши результат обрахунку кодового синдрому S оберіть правильну
відповідь.",
        "correctANS": "1",
        "ansver": [
          {
            "ansver": "Існує помилка."
          },
```

```
            {
              "ansver": "Помилка відсутня."
            }
          ],
          "hint": [
            {
              "hintQeNumType": "1",
              "hintNumQe": "9"
            }
          ]
        },
        {
          "qeNumQue": "5",
          "regime": "pr",
          "question": "img",
          "correctANSStr": "0,125",
          "hint": [
            {
              "hintQeNumType": "1",
              "hintNumQe": "10"
            }
          ]
        }
      ]
    },
    {
      "qeTypeNum": "1",
      "qeTypeName": "Код з перевіркою на непарність",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на непарність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [
        {
          "qeNumQue": "0",
          "question": "Скільки перевірних елементів має код з перевіркою на непарність (r=)?",
          "correctANS": "3",
          "ansver": [
            {
              "ansver": "r=2"
            },
            {
              "ansver": "r=-1"
            },
            {
              "ansver": "r=1"
            },
            {
              "ansver": "r=-3"
            }
          ]
        },
        {
          "qeNumQue": "1",
          "regime": "test",
          "question": "Скільки інформаційних елементів має код з перевіркою на непарність (k=)?",
          "correctANS": "1",
          "ansver": [
            {
              "ansver": "k=(n-1)"
            },
            {
              "ansver": "k=(r-1)"
            },
            {
              "ansver": "k=(n-2)"
            },
            {
              "ansver": "k=(n+n-1)"
```

```
            }
          ]
        },
        {
          "qeNumQue": "2",
          "regime": "test",
          "question": "Як позначається код з перевіркою на непарність?",
          "correctANS": "3",
          "ansver": [
            {
              "ansver": "(k,n-2)"
            },
            {
              "ansver": "(r,k-1)"
            },
            {
              "ansver": "(n,n-1)"
            },
            {
              "ansver": "(n,n-4)"
            }
          ]
        },
        {
          "qeNumQue": "3",
          "regime": "test",
          "question": "Який вигляд має формула для створення перевірного елементу в коді з
перевіркою на непарність?",
          "correctANS": "1",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "4",
          "regime": "test",
          "question": "Як утворюється кодова комбінація в коді з перевіркою на непарність?",
          "correctANS": "2",
          "ansver": [
            {
              "ansver": "Кодова комбінація утворюється доповненням комбінації k – елементного
первинного коду одним елементом таким чином, щоб кількість одиниць у новому n – розрядному (n = k
+1) коді була парною."
            },
            {
              "ansver": "Кожна комбінація має непарну кількість одиниць, тобто додатковий перевірний
елемент формують, виходячи з кількості одиниць у початковій кодовій комбінації; при парній кількості
перевірний елемент дорівнює одиниці, а при непарній – нулю."
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "5",
          "regime": "test",
```

```
  "question": "img",
  "correctANS": "4",
  "ansver": [
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    }
  ]
},
{
  "qeNumQue": "6",
  "regime": "test",
  "question": "img",
  "correctANS": "3",
  "ansver": [
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    }
  ]
},
{
  "qeNumQue": "7",
  "regime": "test",
  "question": "Про відсутність помилки в комбінації свідчить умова?",
  "correctANS": "2",
  "ansver": [
    {
      "ansver": "S=0"
    },
    {
      "ansver": "S=1"
    },
    {
      "ansver": "S=2"
    },
    {
      "ansver": "S=3"
    }
  ]
},
{
  "qeNumQue": "8",
  "regime": "test",
  "question": "Яким виразом визначається надмірність коду з перевіркою на непарність?",
  "correctANS": "3",
  "ansver": [
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
```

```
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        }
      ],
      "questionDetailsPr": [
        {
          "qeNumQue": "0",
          "regime": "pr",
          "question": "Побудуйте кодову комбінацію кодом з перевіркою на непарність.",
          "correctANSStr": "01101101",
          "hint": [
            {
              "hintQeNumType": "1",
              "hintNumQe": "5"
            }
          ]
        },
        {
          "qeNumQue": "1",
          "regime": "pr",
          "question": "Розрахуйте довжину кодової комбінації (n=)",
          "correctANSStr": "8",
          "hint": [
            {
              "hintQeNumType": "1",
              "hintNumQe": "4"
            }
          ]
        },
        {
          "qeNumQue": "2",
          "regime": "pr",
          "question": "img",
          "correctANSStr": "11101101",
          "hint": [
            {
              "hintQeNumType": "0",
              "hintNumQe": "0"
            }
          ]
        },
        {
          "qeNumQue": "3",
          "regime": "pr",
          "question": "Розрахуйте кодовий синдром (S=)",
          "correctANSStr": "1",
          "hint": [
            {
              "hintQeNumType": "1",
              "hintNumQe": "8"
            }
          ]
        },
        {
          "qeNumQue": "4",
          "regime": "test",
          "question": "Використавши результат обрахунку кодового синдрому S оберіть правильну
відповідь.",
          "correctANS": "1",
          "ansver": [
            {
              "ansver": "Існує помилка."
```

```
            },
            {
              "ansver": "Помилка відсутня."
            }
          ],
          "hint": [
            {
              "hintQeNumType": "2",
              "hintNumQe": "8"
            }
          ]
        },
        {
          "qeNumQue": "5",
          "regime": "pr",
          "question": "img",
          "correctANSStr": "0,125",
          "hint": [
            {
              "hintQeNumType": "2",
              "hintNumQe": "9"
            }
          ]
        }
      ]
    },
    {
      "qeTypeNum": "2",
      "qeTypeName": "Код із простим повторенням",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами двійковими кодами з простим повторенням, виявити однократну помилку та визначити надмірність
коду.",
      "questionDetails": [
        {
          "qeNumQue": "0",
          "question": "Скільки перевірних елементів має код із простим повторенням (r=)?",
          "correctANS": "1",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "1",
          "regime": "test",
          "question": "Як утворюється кодова комбінація в коді з простим повторенням?",
          "correctANS": "3",
          "ansver": [
            {
              "ansver": "Кодова комбінація утворюється доповненням комбінації k – елементного
первинного коду одним елементом таким чином, щоб кількість одиниць у новому n – розрядному (n = k
+1) коді була парною."
            },
            {
              "ansver": "Кожна комбінація має непарну кількість одиниць, тобто додатковий перевірний
елемент формують, виходячи з кількості одиниць у початковій кодовій комбінації; при парній кількості
перевірний елемент дорівнює одиниці, а при непарній – нулю."
            },
```

```
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "2",
          "regime": "test",
          "question": "img",
          "correctANS": "2",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "3",
          "regime": "test",
          "question": "Яким методом перевіряють наявність помилок у закодованій комбінації кодом із
простим повторенням?",
          "correctANS": "1",
          "ansver": [
            {
              "ansver": "Потрібно провести порівняння однойменних інформаційних і перевірних
елементів у прийнятій кодовій комбінації. Незбіг їх свідчить про наявність помилок у ній."
            },
            {
              "ansver": "Декодування кодової комбінації у декодері ведуть тактами по два елементи у
кожному такті. При цьому два елементи одного такту не повинні мати однакове значення, тобто не
повинно бути сполучень «00» та «11»."
            }
          ]
        },
        {
          "qeNumQue": "4",
          "regime": "test",
          "question": "Яким виразом визначається надмірність коду з перевіркою на непарність?",
          "correctANS": "4",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        }
      ],
      "questionDetailsPr": [
        {
          "qeNumQue": "0",
          "regime": "pr",
```

```
          "question": "Закодуйте двійкову комбінацію кодом з простим повторенням.",
          "correctANSStr": "01101100110110",
          "hint": [
            {
              "hintQeNumType": "3",
              "hintNumQe": "2"
            }
          ]
        },
        {
          "qeNumQue": "1",
          "regime": "pr",
          "question": "Розрахуйте довжину кодової комбінації для заданої в прикладі двійкової
комбінації (n=)",
          "correctANSStr": "14",
          "hint": [
            {
              "hintQeNumType": "1",
              "hintNumQe": "4"
            }
          ]
        },
        {
          "qeNumQue": "2",
          "regime": "pr",
          "question": "img",
          "correctANSStr": "01101110110110",
          "hint": [
            {
              "hintQeNumType": "0",
              "hintNumQe": "0"
            }
          ]
        },
        {
          "qeNumQue": "3",
          "regime": "pr",
          "question": "img",
          "correctANSStr": "1",
          "hint": [
            {
              "hintQeNumType": "3",
              "hintNumQe": "4"
            }
          ]
        },
        {
          "qeNumQue": "5",
          "regime": "pr",
          "question": "img",
          "correctANSStr": "0,5",
          "hint": [
            {
              "hintQeNumType": "3",
              "hintNumQe": "5"
            }
          ]
        }
      ]
    },
    {
      "qeTypeNum": "3",
      "qeTypeName": "Інверсний код (із повторенням та інверсією)",
      "qestionForPr": "Закодувати комбінацію 110010 двійкового простого коду (k=6) інверсним кодом
(із повторенням та інверсією), виявити однократну помилку та визначити надмірність коду.",
      "questionDetails": [
        {
          "qeNumQue": "0",
```

```
  "question": "Скільки перевірних елементів має інверсний код (r=)?",
  "correctANS": "4",
  "ansver": [
    {
      "ansver": "r=1"
    },
    {
      "ansver": "r=k+1"
    },
    {
      "ansver": "r=k-1"
    },
    {
      "ansver": "r=k"
    }
  ]
},
{
  "qeNumQue": "1",
  "regime": "test",
  "question": "img",
  "correctANS": "1",
  "ansver": [
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    }
  ]
},
{
  "qeNumQue": "2",
  "regime": "test",
  "question": "img",
  "correctANS": "2",
  "ansver": [
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    }
  ]
},
{
  "qeNumQue": "3",
  "regime": "test",
  "question": "img",
  "correctANS": "3",
  "ansver": [
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    }
  ]
},
{
  "qeNumQue": "4",
  "regime": "test",
```

```
          "question": "Як виявляють наявність помилок на приймальному боці у послідовності?",
          "correctANS": "2",
          "ansver": [
            {
              "ansver": "Потрібно провести порівняння однойменних інформаційних і перевірних
елементів у прийнятій кодовій комбінації. Незбіг їх свідчить про наявність помилок у ній."
            },
            {
              "ansver": "Спочатку підсумовують одиниці, які знаходяться в перших k елементах. Якщо
їх кількість парна, то решту k елементів приймають у позитиві. Обидві зареєстровані частини
комбінацій поелементно порівнюють. За наявності хоча б одного незбігу вся послідовність елементів
бракується. Якщо кількість одиниць серед перших k елементів непарна, то решту k елементів приймають
у негативі (інвертують), після чого поелементно порівнюють їх. Наявність незбігу призводить до
відбраковування всіх 2k елементів."
            },
            {
              "ansver": "Декодування кодової комбінації у декодері ведуть тактами по два елементи у
кожному такті. При цьому два елементи одного такту не повинні мати однакове значення, тобто не
повинно бути сполучень «00» та «11»."
            }
          ]
        },
        {
          "qeNumQue": "5",
          "regime": "test",
          "question": "Яким виразом визначається надмірність інверсного коду?",
          "correctANS": "4",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        }
      ],
      "questionDetailsPr": [
        {
          "qeNumQue": "0",
          "regime": "pr",
          "question": "Закодуйте двійкову комбінацію інверсним кодом.",
          "correctANSStr": "110010001101",
          "hint": [
            {
              "hintQeNumType": "4",
              "hintNumQe": "2"
            },
            {
              "hintQeNumType": "4",
              "hintNumQe": "3"
            }
          ]
        },
        {
          "qeNumQue": "1",
          "regime": "pr",
          "question": "Розрахуйте довжину кодової комбінації для заданої в прикладі двійкової
комбінації (n=)",
          "correctANSStr": "12",
          "hint": [
            {
```

```
            "hintQeNumType": "1",
            "hintNumQe": "4"
          }
        ]
      },
      {
        "qeNumQue": "2",
        "regime": "pr",
        "question": "img",
        "correctANSStr": "110000001101",
        "hint": [
          {
            "hintQeNumType": "0",
            "hintNumQe": "0"
          }
        ]
      },
      {
        "qeNumQue": "4",
        "regime": "test",
        "question": "img",
        "correctANS": "1",
        "ansver": [
          {
            "ansver": "Існує помилка."
          },
          {
            "ansver": "Помилка відсутня."
          }
        ],
        "hint": [
          {
            "hintQeNumType": "4",
            "hintNumQe": "5"
          }
        ]
      },
      {
        "qeNumQue": "5",
        "regime": "pr",
        "question": "img",
        "correctANSStr": "0,5",
        "hint": [
          {
            "hintQeNumType": "4",
            "hintNumQe": "6"
          }
        ]
      }
    ]
  },
  {
    "qeTypeNum": "4",
    "qeTypeName": "Кореляційний код",
    "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити надмірність коду.",
    "questionDetails": [

    ]
  },
  {
    "qeTypeNum": "5",
    "qeTypeName": "Код із сталою (постійною) вагою",
    "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити надмірність коду.",
    "questionDetails": [
```

```
      ]
    },
    {
      "qeTypeNum": "6",
      "qeTypeName": "Код із кількістю одиниць у комбінації, кратною 3",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
    },
    {
      "qeTypeNum": "7",
      "qeTypeName": "Код з перевіркою за модулем q",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
    }
  ]
}
```


## Б.2 Вміст файлу «ru_RU.json»


```
{
  "items": [
    {
      "key": "null",
      "value": "Пустое поле"
    },
    {
      "key": "ButLangOk",
      "value": "Выбрать"
    },
    {
      "key": "ButNextQe",
      "value": "Следущий вопрос"
    },
    {
      "key": "ButEXIT",
      "value": "Выход"
    },
    {
      "key": "ButHELP",
      "value": "Помощь"
    },
    {
      "key": "ButOK",
      "value": "Ответить"
    },
    {
      "key": "ButStart",
      "value": "Начать"
    },
    {
      "key": "ButLang",
      "value": "Язык"
    },
    {
      "key": "ButStop",
      "value": "Закончить"
    },
    {
      "key": "TextTopic",
```

```
      "value": "Вас приветствует тренажер с темы\n\"Коды с обнаружением ошибок\""
    },
    {
      "key": "TextTooltip",
      "value": "Для начала тренировки выберите язык и нажмите кнопку начать \"Начать\""
    },
    {
      "key": "TextForAutor",
      "value": "Автор работы: \n студент группы КН м 21 \n Жаворонок Яролслав \n\n Научный
руководитель: \n к. Ф.-м. н., доцент \nТ. А. Парфенова"
    }
  ],
  "question": [
    {
      "qeTypeNum": "0",
      "qeTypeName": "Код из проверкой на парность",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [
        {
          "qeNumQue": "0",
          "question": "Сколько проверочных элементов (r) имеет код с проверкой на четность?",
          "correctANS": "2",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "1",
          "question": "Скільки інформаційних елементів (k) має код з перевіркою на парність?",
          "correctANS": "1",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "2",
          "question": "Як позначається код з перевіркою на парність?",
          "correctANS": "3",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
```

```
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "3",
          "question": "Параметр (n) обраховується за формулою?",
          "correctANS": "3",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "4",
          "question": "Який вигляд має формула для створення перевірного елементу в коді з
перевіркою на парність?",
          "correctANS": "1",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "5",
          "question": "Як утворюється кодова комбінація в коді з перевіркою на парність?",
          "correctANS": "1",
          "ansver": [
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            },
            {
              "ansver": "img"
            }
          ]
        },
        {
          "qeNumQue": "6",
          "question": "img",
          "correctANS": "4",
```

```
  "ansver": [
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    }
  ]
},
{
  "qeNumQue": "7",
  "question": "img",
  "correctANS": "3",
  "ansver": [
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    }
  ]
},
{
  "qeNumQue": "8",
  "question": "Відсутність помилки в комбінації вважається коли? ",
  "correctANS": "1",
  "ansver": [
    {
      "ansver": "S=0"
    },
    {
      "ansver": "S=1"
    },
    {
      "ansver": "S=-1"
    },
    {
      "ansver": "S=2"
    }
  ]
},
{
  "qeNumQue": "9",
  "question": "Яким виразом визначається надмірність коду?",
  "correctANS": "3",
  "ansver": [
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
      "ansver": "img"
    },
    {
```

```
            "ansver": "img"
          }
        ]
      }
    ],
    "questionDetailsPr": [
      {
        "qeNumQue": "0",
        "question": "Побудуйте кодову комбінацію кодом з перевіркою на парність",
        "correctANSStr": "01101100",
        "hint": [
          {
            "hintQeNumType": "1",
            "hintNumQe": "5"
          },
          {
            "hintQeNumType": "1",
            "hintNumQe": "6"
          }
        ]
      },
      {
        "qeNumQue": "1",
        "question": "Розрахуйте довжину кодової комбінації (n=)",
        "correctANSStr": "8",
        "hint": [
          {
            "hintQeNumType": "1",
            "hintNumQe": "4"
          }
        ]
      },
      {
        "qeNumQue": "2",
        "question": "img",
        "correctANSStr": "01001100",
        "hint": [
          {
            "hintQeNumType": "0",
            "hintNumQe": "0"
          }
        ]
      },
      {
        "qeNumQue": "3",
        "question": "Розрахуйте кодовий синдром (S= )",
        "correctANSStr": "1",
        "hint": [
          {
            "hintQeNumType": "1",
            "hintNumQe": "8"
          }
        ]
      },
      {
        "qeNumQue": "4",
        "question": "Використавши результат обрахунку кодового синдрому S оберіть правильну
відповідь.",
        "correctANS": "1",
        "ansver": [
          {
            "ansver": "Існує помилка."
          },
          {
            "ansver": "Помилка відсутня."
          }
        ],
        "hint": [
```

```
            {
              "hintQeNumType": "1",
              "hintNumQe": "9"
            }
          ]
        },
        {
          "qeNumQue": "5",
          "question": "img",
          "correctANSStr": "0,125",
          "hint": [
            {
              "hintQeNumType": "1",
              "hintNumQe": "10"
            }
          ]
        }
      ]
    },
    {
      "qeTypeNum": "2",
      "qeTypeName": "Код с простым повторением",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
    },
    {
      "qeTypeNum": "3",
      "qeTypeName": "Инверсный код (с повторением и инверсией)",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
    },
    {
      "qeTypeNum": "4",
      "qeTypeName": "Корреляционный код",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
    },
    {
      "qeTypeNum": "5",
      "qeTypeName": "Код с постоянным весом",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
    },
    {
      "qeTypeNum": "6",
      "qeTypeName": "Код с количеством единиц в комбинации, кратной 3",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
```

```
    },
    {
      "qeTypeNum": "7",
      "qeTypeName": "Код с проверкой по модулю q",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити надмірність коду.",
      "questionDetails": [

      ]
    }
  ]
}
```

## Б.3 Вміст файлу «en_US.json»

```
{
  "items": [
    {
      "key": "null",
      "value": "Empty field"
    },
    {
      "key": "ButLangOk",
      "value": "Choose"
    },
    {
      "key": "ButNextQe",
      "value": "Next question"
    },
    {
      "key": "ButEXIT",
      "value": "Exit"
    },
    {
      "key": "ButHELP",
      "value": "Help"
    },
    {
      "key": "ButOK",
      "value": "Answer"
    },
    {
      "key": "ButStart",
      "value": "Start"
    },
    {
      "key": "ButLang",
      "value": "Language"
    },
    {
      "key": "ButStop",
      "value": "Finish"
    },
    {
      "key": "TextTopic",
      "value": "You are greeted by a simulator on the topic\n\"Error detection codes\""
    },
    {
      "key": "TextTooltip",
      "value": "To start training, select the language and click the start button \"Start\""
    },
    {
      "key": "TextForAutor",
      "value": "Author of the work:\nstudent of the group KN m-21\nZhaivoronok Yarolslav\n\n Supervisor: к. ф.-м. н., доцент\n T. O. Parfyonova"
    }
```

```
  ],
  "question": [
    {
        "qeTypeNum": "0",
        "qeTypeName": "Parity check code",
        "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
        "questionDetails": [
          {
            "qeNumQue": "0",
            "question": "How many parity check elements (r) does the parity code have?",
            "correctANS": "2",
            "ansver": [
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              }
            ]
          },
          {
            "qeNumQue": "1",
            "question": "Скільки інформаційних елементів (k) має код з перевіркою на парність?",
            "correctANS": "1",
            "ansver": [
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              }
            ]
          },
          {
            "qeNumQue": "2",
            "question": "Як позначається код з перевіркою на парність?",
            "correctANS": "3",
            "ansver": [
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              }
            ]
          },
          {
            "qeNumQue": "3",
```

```
            "question": "Параметр (n) обраховується за формулою?",
            "correctANS": "3",
            "ansver": [
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              }
            ]
          },
          {
            "qeNumQue": "4",
            "question": "Який вигляд має формула для створення перевірного елементу в коді з
перевіркою на парність?",
            "correctANS": "1",
            "ansver": [
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              }
            ]
          },
          {
            "qeNumQue": "5",
            "question": "Як утворюється кодова комбінація в коді з перевіркою на парність?",
            "correctANS": "1",
            "ansver": [
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              }
            ]
          },
          {
            "qeNumQue": "6",
            "question": "img",
            "correctANS": "4",
            "ansver": [
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
                "ansver": "img"
              },
              {
```

```
        "ansver": "img"
      }
    ]
  },
  {
    "qeNumQue": "7",
    "question": "img",
    "correctANS": "3",
    "ansver": [
      {
        "ansver": "img"
      },
      {
        "ansver": "img"
      },
      {
        "ansver": "img"
      },
      {
        "ansver": "img"
      }
    ]
  },
  {
    "qeNumQue": "8",
    "question": "Відсутність помилки в комбінації вважається коли? ",
    "correctANS": "1",
    "ansver": [
      {
        "ansver": "S=0"
      },
      {
        "ansver": "S=1"
      },
      {
        "ansver": "S=-1"
      },
      {
        "ansver": "S=2"
      }
    ]
  },
  {
    "qeNumQue": "9",
    "question": "Яким виразом визначається надмірність коду?",
    "correctANS": "3",
    "ansver": [
      {
        "ansver": "img"
      },
      {
        "ansver": "img"
      },
      {
        "ansver": "img"
      },
      {
        "ansver": "img"
      }
    ]
  }
],
"questionDetailsPr": [
  {
    "qeNumQue": "0",
    "question": "Побудуйте кодову комбінацію кодом з перевіркою на парність",
    "correctANSStr": "01101100",
    "hint": [
```

```
            {
              "hintQeNumType": "1",
              "hintNumQe": "5"
            },
            {
              "hintQeNumType": "1",
              "hintNumQe": "6"
            }
          ]
        },
        {
          "qeNumQue": "1",
          "question": "Розрахуйте довжину кодової комбінації (n=)",
          "correctANSStr": "8",
          "hint": [
            {
              "hintQeNumType": "1",
              "hintNumQe": "4"
            }
          ]
        },
        {
          "qeNumQue": "2",
          "question": "img",
          "correctANSStr": "01001100",
          "hint": [
            {
              "hintQeNumType": "0",
              "hintNumQe": "0"
            }
          ]
        },
        {
          "qeNumQue": "3",
          "question": "Розрахуйте кодовий синдром (S= )",
          "correctANSStr": "1",
          "hint": [
            {
              "hintQeNumType": "1",
              "hintNumQe": "8"
            }
          ]
        },
        {
          "qeNumQue": "4",
          "question": "Використавши результат обрахунку кодового синдрому S оберіть правильну
відповідь.",
          "correctANS": "1",
          "ansver": [
            {
              "ansver": "Існує помилка."
            },
            {
              "ansver": "Помилка відсутня."
            }
          ],
          "hint": [
            {
              "hintQeNumType": "1",
              "hintNumQe": "9"
            }
          ]
        },
        {
          "qeNumQue": "5",
          "question": "img",
          "correctANSStr": "0,125",
          "hint": [
```

```
            {
              "hintQeNumType": "1",
              "hintNumQe": "10"
            }
          ]
        }
      ]
    },
    {
      "qeTypeNum": "2",
      "qeTypeName": "Code with simple repetition",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
    },
    {
      "qeTypeNum": "3",
      "qeTypeName": "Inverse code (with repetition and inversion)",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
    },
    {
      "qeTypeNum": "4",
      "qeTypeName": "Correlation code",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
    },
    {
      "qeTypeNum": "5",
      "qeTypeName": "Code with constant (constant) weight",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
    },
    {
      "qeTypeNum": "6",
      "qeTypeName": "Code with the number of units in the combination, a multiple of 3",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
    },
    {
      "qeTypeNum": "7",
      "qeTypeName": "Code with verification modulo q",
      "qestionForPr": "Закодувати комбінацію 0110110 двійкового простого коду (k=7) двійковими
кодами, що виявляють помилки з перевіркою на парність, виявити однократну помилку та визначити
надмірність коду.",
      "questionDetails": [

      ]
    }
```

```
  ]
}
```

## Б.4 Код скрипту «BtnSwitchLang»

```
using UnityEngine;

public class BtnSwitchLang : MonoBehaviour
{
    [SerializeField]
    private LocalizationManager localizationManager;
    public BtnSwitchLang(LocalizationManager localizationManager)
    {
        this.localizationManager = localizationManager;
    }
    public void OnButtonClick()
    {
        localizationManager.CurrentLanguage = name;
    }
}
```

## Б.5 Код скрипту «LocalizationData»

```
[System.Serializable]
public class LocalizationData
{
    public LocalizationItem[] items;
    public QuestionInfo[] question;
}

[System.Serializable]
public class LocalizationItem
{
    public string key;
    public string value;
}

[System.Serializable]
public class QuestionItem
{
    public string regime;
    public int qeNumQue;
    public string question;
    public Ansver[] ansver;
    public Hint[] hint;
    public int correctANS;

    public string correctANSStr;
}

[System.Serializable]
public class QuestionInfo
{
    public string qeTypeName;
    public string qestionForPr;
    public int qeTypeNum;
    public QuestionItem[] questionDetails;
    public QuestionItem[] questionDetailsPr;
}

[System.Serializable]
public class Ansver
{
```

```
    public string ansver;

}
public class Hint
{
    public string hintQeNumType;
    public string hintNumQe;
}
```

## Б.6 Код скрипту «LocalizationManager»

```
using System;
using System.IO;
using System.Collections.Generic;
using UnityEngine.Networking;
using UnityEngine;
using System.Collections;

public class LocalizationManager : MonoBehaviour
{
    public static string currentLanguage;
    private Dictionary<string, string> localizedText;
    public static bool isReady = false;
    public LocalizationData loadedData;
    public delegate void ChangeLangText();
    public event ChangeLangText OnLanguageChanged;

    void Awake()
    {
        if (!PlayerPrefs.HasKey("Language"))
        {

            if (Application.systemLanguage == SystemLanguage.Russian || Application.systemLanguage
== SystemLanguage.Belarusian)
            {
                PlayerPrefs.SetString("Language", "ru_RU");
            }else
            if (Application.systemLanguage == SystemLanguage.Ukrainian)
            {
                PlayerPrefs.SetString("Language", "ua_UA");
            }
            else
            {
                PlayerPrefs.SetString("Language", "en_US");
            }
        }
        currentLanguage = PlayerPrefs.GetString("Language");

        LoadLocalizedText(currentLanguage);
    }

    IEnumerator LoadTextFromServer(string url)
    {
        UnityWebRequest request = UnityWebRequest.Get(url);

        yield return request.SendWebRequest();
        loadedData = JsonUtility.FromJson<LocalizationData>(request.downloadHandler.text);

        request.Dispose();
    }
    public void LoadLocalizedText(string langName)
    {
        string path = Application.streamingAssetsPath + "/Languages/" + langName + ".json";
```

```
    string dataAsJson;
    /*
    #if UNITY_ANDROID && !UNITY_EDITOR
                WWW reader = new WWW(path);
                while (!reader.isDone) { }

                dataAsJson = reader.text;

    #else
            dataAsJson = File.ReadAllText(path);
    #endif
    */
    //loadedData = JsonUtility.FromJson<LocalizationData>(dataAsJson);

    TextAsset file = Resources.Load("Languages/" + langName) as TextAsset;
    string content = file.ToString();

    loadedData = JsonUtility.FromJson<LocalizationData>(content);
    //!!!!!!!!!!!!!!!!!!!!!!!!!!!
    localizedText = new Dictionary<string, string>();
    for (int i = 0; i < loadedData.items.Length; i++)
    {
        localizedText.Add(loadedData.items[i].key, loadedData.items[i].value);
    }

    PlayerPrefs.SetString("Language", langName);
    currentLanguage = PlayerPrefs.GetString("Language");
    isReady = true;

    OnLanguageChanged?.Invoke();
}

public string GetLocalizedValue(string key)
{
    if (localizedText.ContainsKey(key))
    {
        return localizedText[key];
    }
    else
    {
        throw new Exception("Localized text with key \"" + key + "\" not found");
    }
}

public string CurrentLanguage
{
    get
    {
        return currentLanguage;
    }
    set
    {
        PlayerPrefs.SetString("Language", value);
        currentLanguage = PlayerPrefs.GetString("Language");
        LoadLocalizedText(currentLanguage);
    }
}

public void OnButtonClick()
{
    currentLanguage = name;

}
public bool IsReady
{
    get
    {
        return isReady;
```

```
        }
    }
}
```

## Б.7 Код скрипту «LocalizedText»

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

public class LocalizedText : MonoBehaviour
{
    [SerializeField]
    private string key="null";

    private LocalizationManager localizationManager;
    private Text text;


    void Awake()
    {
        if (localizationManager == null)
        {
            localizationManager =
GameObject.FindGameObjectWithTag("LocalizationManager").GetComponent<LocalizationManager>();
        }
        if (text == null)
        {
            text = GetComponent<Text>();
        }
        localizationManager.OnLanguageChanged += UpdateText;
    }

    void Start()
    {
        UpdateText();
    }

    private void OnDestroy()
    {
        localizationManager.OnLanguageChanged -= UpdateText;
    }

    virtual protected void UpdateText()
    {
        if (gameObject == null) return;

        if (localizationManager == null)
        {
            localizationManager =
GameObject.FindGameObjectWithTag("LocalizationManager").GetComponent<LocalizationManager>();
        }
        if (text == null)
        {
            text = GetComponent<Text>();
        }

        text.text = localizationManager.GetLocalizedValue(key);
    }
}
```

## Б.8 Код скрипту «TaskControl»

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

public class TaskControl : MonoBehaviour
{
    private Loader loader;
    public Text nameTest;
    public Text ballsTest;
    int numTask;

    void Awake()
    {
        if (loader == null)
        {
            loader = GameObject.FindGameObjectWithTag("LocalizationManager").GetComponent<Loader>();
        }

    }
    public void TaskControls(string name, string balls,int numTask)
    {
        nameTest.text = name;
        ballsTest.text = balls;
        this.numTask= numTask;
    }

    public void LoadTask()
    {
        loader.StartSetRegime(numTask);
    }
}
```

## Б.9 Код скрипту «TaskToMenu»

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

public class TaskToMenu : MonoBehaviour
{
    private LocalizationManager localizationManager;

    public GameObject prefabTask;
    public List<GameObject> tasks;

    void AddNewTask(string taskName,string balls,int numQestion)
    {
        tasks.Add(Instantiate(prefabTask, this.transform.position, this.transform.rotation =
Quaternion.identity, transform));
        tasks[tasks.Count-1].GetComponent<TaskControl>().TaskControls(taskName, balls, numQestion);
    }

    void Awake()
    {
        if (localizationManager == null)
        {
            localizationManager =
GameObject.FindGameObjectWithTag("LocalizationManager").GetComponent<LocalizationManager>();
```

```
        }
    }
    void Start ()
    {
        for (int i = 0; i < localizationManager.loadedData.question.Length; i++)
        {
            AddNewTask(localizationManager.loadedData.question[i].qeTypeName, "7/15",
localizationManager.loadedData.question[i].qeTypeNum);
        }
    }
}
```

## Б.10 Код скрипту «Loader»

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

public class Loader : MonoBehaviour
{
    public GameObject Menu;
    public GameObject PreafTesting;
    public GameObject TaskList;
    public GameObject RegimeBtt;

    public List<GameObject> tasks;

    public bool itsTeory;


    public GameObject exitButton;
    public GameObject menuText;
    public GameObject regimeText;
    private LocalizationManager localizationManager;
    public TaskList taskList;
    public EndWindow endWindow;

    public int numQe = 0; // Номер зашального типу питання
    public int numActiveQe = 0; // Номер активного питання в типі
    public int numTask = 0; // Номер питання в типі
    public int bals = 0;
    public int EndTaskCount = 0;

    public void StrRegimeTeory()
    {
        itsTeory = true;
    }

    public void StrRegimePrakt()
    {
        itsTeory = false;
    }


    void Awake()
    {
        if (localizationManager == null)
        {
            localizationManager =
GameObject.FindGameObjectWithTag("LocalizationManager").GetComponent<LocalizationManager>();
```

```
        }
        if (taskList == null)
        {
            taskList = GameObject.FindGameObjectWithTag("ContentMenu").GetComponent<TaskList>();
        }

    }


    private void Start()
    {

        Menu.SetActive(true);
        endWindow.UnloadEndWindow();
    }

    public void StartSetRegime(int numTask) {
        menuText.SetActive(false);
        regimeText.SetActive(true);
        Debug.Log("Вибір режиму дял типу"+ numTask);
        RegimeBtt.SetActive(true);
        Menu.SetActive(false);
        this.numQe = numTask;

    }
    public void StartTask()
    {
        regimeText.SetActive(false);
        exitButton.SetActive(false);
        Debug.Log("StartTask");
        RegimeBtt.SetActive(false);
        InstAllTaskType(numQe);
        TaskToTaskList(numQe);
        ActiveQe(0);
    }

    public void InstAllTaskType(int numQuestionTupe)
    {
        Debug.Log("InstAllTaskType" + numQuestionTupe);
        if (itsTeory)
        {
            for (int i = 0; i <
localizationManager.loadedData.question[numQuestionTupe].questionDetails.Length; i++)
            {
                numTask = i;
                tasks.Add(Instantiate(PreafTesting, transform));
            }

        }
        else
        {
            for (int i = 0; i <
localizationManager.loadedData.question[numQuestionTupe].questionDetailsPr.Length; i++)
            {
                numTask = i;
                tasks.Add(Instantiate(PreafTesting, transform));
            }
        }
        foreach (GameObject game in tasks)
        {
            game.SetActive(false);
        }

    }


    public void TaskToTaskList(int numQuestionTupe)
    {
```

```
        Debug.Log("TaskToTaskList" + numQuestionTupe);
        TaskList.GetComponentInChildren<TaskList>().AddNewNumberTask(numQuestionTupe,itsTeory);
    }


    void DestroyAllQuestion()
    {
        foreach(GameObject game in tasks)
        {
            Destroy(game);
        }
        tasks.Clear();
        numQe = 0;
        numActiveQe = 0;
        numTask = 0;
        bals = 0;
        EndTaskCount = 0;

}

    public void ActiveQe(int qeToActive)
    {
        Debug.Log("TActiveQet" + qeToActive);
        tasks[numActiveQe].SetActive(false);
        numActiveQe = qeToActive;
        tasks[numActiveQe].SetActive(true);

    }

    public void ActiveNextQe()
    {
        tasks[numActiveQe].SetActive(false);

        if (itsTeory == true)
        {
            if (EndTaskCount ==
localizationManager.loadedData.question[numQe].questionDetails.Length)
            {
                LoadRezult();
            }
            else if (numActiveQe >=
localizationManager.loadedData.question[numQe].questionDetails.Length-1)
            {
                numActiveQe = 0;
                ActiveQe(numActiveQe);
            }
            else
            {
                numActiveQe++;
                tasks[numActiveQe].SetActive(true);
            }
        }
        else
        {
            if (EndTaskCount ==
localizationManager.loadedData.question[numQe].questionDetailsPr.Length)
            {
                LoadRezult();
            }
            else if (numActiveQe >=
localizationManager.loadedData.question[numQe].questionDetailsPr.Length-1)
            {
                numActiveQe = 0;
                ActiveQe(numActiveQe);
            }
            else
            {
                numActiveQe++;
```

```
                tasks[numActiveQe].SetActive(true);
            }
        }

    }

    public void LoadRezult()
    {
        tasks[numActiveQe].SetActive(false);
        endWindow.LoadEndWindow();
        foreach (GameObject game in tasks)
        {
            game.GetComponent<ManageQuestion>().ButtonsOff();
        }

    }

    public void LoadMenu()
    {
        Menu.SetActive(true);
        endWindow.UnloadEndWindow();
        DestroyAllQuestion();
        taskList.DestroyAllTasks();
        exitButton.SetActive(true);
        menuText.SetActive(true);
    }
}
```

## Б.11 Код скрипту «TaskList»

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

public class TaskList : MonoBehaviour
{
    public GameObject prefabTask;
    public List<GameObject> tasks;
    public Image imgTask;
    private LocalizationManager localizationManager;
    void Awake()
    {
        if (localizationManager == null)
        {
            localizationManager =
GameObject.FindGameObjectWithTag("LocalizationManager").GetComponent<LocalizationManager>();
        }
    }
    public void AddNewNumberTask(int taskNum, bool itsTeory)
    {
        DestroyAllTasks();
        Debug.Log("AddNewNumberTask " + taskNum + " чек" + itsTeory);
        if (itsTeory)
        {
            for (int i = 0; i <
localizationManager.loadedData.question[taskNum].questionDetails.Length; i++)
            {
                tasks.Add(Instantiate(prefabTask, this.transform.position, this.transform.rotation =
Quaternion.identity, transform));
                tasks[tasks.Count - 1].GetComponent<ListTaskControlName>().TaskControls(i + 1);
            }
        }
        else
```

```
        {
            for (int i = 0; i <
localizationManager.loadedData.question[taskNum].questionDetailsPr.Length; i++)
            {
                tasks.Add(Instantiate(prefabTask, this.transform.position, this.transform.rotation =
Quaternion.identity, transform));
                tasks[tasks.Count - 1].GetComponent<ListTaskControlName>().TaskControls(i + 1);
            }
        }
    }
    public void DestroyAllTasks()
    {
        foreach (GameObject game in tasks)
        {
            Destroy(game);
        }
        tasks.Clear();
    }
    public void TrueAnsver(int num)
    {
        tasks[num].GetComponent<Image>().color = new Color(0f, 1f, 0f, 1f);
        tasks[num].GetComponent<ListTaskControlName>().SetColor(new Color(0f, 1f, 0f, 1f));
    }
    public void FalseAnsver(int num)
    {
        tasks[num].GetComponent<Image>().color = new Color(1f, 0f, 0f, 1f);
        tasks[num].GetComponent<ListTaskControlName>().SetColor(new Color(1f, 0f, 0f, 1f));
    }
}
```

## Б.12 Код скрипту «ListTaskControlName»

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

public class ListTaskControlName : MonoBehaviour
{

    public Text numberTest;
    public int numTest;
    private Loader loader;
    public Image imgTask;

    void Awake()
    {
        if (loader == null)
        {
            loader = GameObject.FindGameObjectWithTag("LocalizationManager").GetComponent<Loader>();
        }
    }

    public void TaskControls(int numTask)
    {
        numberTest.text = numTask.ToString();
        this.numTest = numTask;
    }

    public void SetVisibleTask()
    {
        loader.ActiveQe(numTest-1);
        if (loader.endWindow.gameObject.activeSelf)
        {
```

```
            loader.endWindow.OffText();
        }
    }
    public void SetColor(Color color)
    {
        imgTask.color = color;
    }
}
```

## Б.13 Код скрипту «ManageQuestion»

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


public class ManageQuestion : MonoBehaviour
{

    public GameObject[] testingTextPlase;
    public GameObject[] testingImgPlase;
    public Text questionText;
    public Image questionImg;

    //Практическая часть
    public GameObject EnterAnsverElement;
    public GameObject Fon;
    public Text PacticeAnver;
    public Text questionTextPr;
    public Image questionImgPr;
    public Image ansImgPr;



    public int numberQuestion;
    public int numberTypeQuestion;
    private LocalizationManager localizationManager;
    private Loader loader;


    Sprite sprite;

    public GameObject BtnOK;
    public GameObject BtnNextQe;

    public ToggleGroup toggleGroup;
    public Toggle[] toggleComponents;



    public Text numQuestion;
    public Text nameQuestion;


    private TaskList taskList;
    public GameObject Buttons;
    public GameObject Ansver;

    List<int> tempObj = new List<int>();
    void Awake()
    {
```

```
        if (localizationManager == null)
        {
            localizationManager =
GameObject.FindGameObjectWithTag("LocalizationManager").GetComponent<LocalizationManager>();
            loader = GameObject.FindGameObjectWithTag("LocalizationManager").GetComponent<Loader>();
        }
        if (taskList == null)
        {
            taskList = GameObject.FindGameObjectWithTag("ContentMenu").GetComponent<TaskList>();
        }
        numberTypeQuestion = loader.numQe;
        numberQuestion = loader.numTask;
        LoadData();
    }

    bool TheckImg(string text)
    {
        if (text == "img") return true;
        else return false;

    }



    void LoadData()
    {
        LoadQuestion();
        TestLoader();
    }

    void LoadQuestion()
    {
        nameQuestion.text = localizationManager.loadedData.question[numberTypeQuestion].qeTypeName;
        numQuestion.text = Convert.ToString(numberQuestion + 1);
        if (loader.itsTeory == true)
        {
            if
(TheckImg(localizationManager.loadedData.question[numberTypeQuestion].questionDetails[numberQuestion
].question))
            {
                questionImg.gameObject.SetActive(true);
                questionText.gameObject.SetActive(false);
                sprite = Resources.Load<Sprite>("Formulas/Theory/" + (numberTypeQuestion + 1) + "/"
+ (numberQuestion + 1) + "/0");
                questionImg.sprite = sprite;
            }
            else
            {
                questionImg.gameObject.SetActive(false);
                questionText.gameObject.SetActive(true);
                questionText.text =
localizationManager.loadedData.question[numberTypeQuestion].questionDetails[numberQuestion].question
;
            }
        }
        else
        {
            questionImg.gameObject.SetActive(false);
            questionText.gameObject.SetActive(true);
            questionText.text =
localizationManager.loadedData.question[numberTypeQuestion].qestionForPr;

        }
    }

    void TestLoader()
    {
        Debug.Log("Загрузка вопросів у питання" + numberTypeQuestion + " " + numberQuestion);
```

```
        if (loader.itsTeory == true)
        {
            for (int i = 0; i <
localizationManager.loadedData.question[numberTypeQuestion].questionDetails[numberQuestion].ansver.L
ength; i++)
            {
                if
(TheckImg(localizationManager.loadedData.question[numberTypeQuestion].questionDetails[numberQuestion
].ansver[i].ansver))
                {
                    testingTextPlase[i].SetActive(false);
                    testingImgPlase[i].SetActive(true);
                    tempObj.Add(i);
                    string puch = "Formulas/Theory/" + (numberTypeQuestion + 1) + "/" +
(numberQuestion + 1) + "/" + (i + 1);
                    sprite = Resources.Load<Sprite>(puch);
                    testingImgPlase[i].GetComponent<QeToImage>().image.GetComponent<Image>().sprite
= sprite;

                }
                else
                {
                    testingTextPlase[i].SetActive(true);
                    testingImgPlase[i].SetActive(false);
                    tempObj.Add(4+i);
                    testingTextPlase[i].GetComponentInChildren<Text>().text =
localizationManager.loadedData.question[numberTypeQuestion].questionDetails[numberQuestion].ansver[i
].ansver;
                }
            }
        }
        else
        {
            if
(localizationManager.loadedData.question[numberTypeQuestion].questionDetailsPr[numberQuestion].regim
e == "test")
            {
                for (int i = 0; i <
localizationManager.loadedData.question[numberTypeQuestion].questionDetailsPr[numberQuestion].ansver
.Length; i++)
                {
                    if
(TheckImg(localizationManager.loadedData.question[numberTypeQuestion].questionDetailsPr[numberQuesti
on].ansver[i].ansver))
                    {
                        testingTextPlase[i].SetActive(false);
                        testingImgPlase[i].SetActive(true);
                        tempObj.Add(i);
                        string puch = "Formulas/Theory/" + (numberTypeQuestion + 1) + "/" +
(numberQuestion + 1) + "/" + (i + 1);
                        sprite = Resources.Load<Sprite>(puch);

testingImgPlase[i].GetComponent<QeToImage>().image.GetComponent<Image>().sprite = sprite;

                    }
                    else
                    {
                        testingTextPlase[i].SetActive(true);
                        testingImgPlase[i].SetActive(false);
                        tempObj.Add(4+i);
                        testingTextPlase[i].GetComponentInChildren<Text>().text =
localizationManager.loadedData.question[numberTypeQuestion].questionDetailsPr[numberQuestion].ansver
[i].ansver;
                    }
                }
            }
            else
            {
```

```
                Fon.SetActive(true);
                if
(TheckImg(localizationManager.loadedData.question[numberTypeQuestion].questionDetailsPr[numberQuesti
on].question))
                {
                    questionTextPr.gameObject.SetActive(false);
                    questionImgPr.gameObject.SetActive(true);
                    string puch = "Formulas/Practice/" + (numberTypeQuestion + 1) + "/" +
(numberQuestion + 1) + "/0";
                    questionImgPr.sprite = Resources.Load<Sprite>(puch);
                }
                else
                {
                    questionTextPr.gameObject.SetActive(true);
                    questionImgPr.gameObject.SetActive(false);
                    questionTextPr.text =
localizationManager.loadedData.question[numberTypeQuestion].questionDetailsPr[numberQuestion].questi
on;
                }
                EnterAnsverElement.SetActive(true);
                ansImgPr.sprite = Resources.Load<Sprite>("Formulas/Practice/" + (numberTypeQuestion
+ 1) + "/" + (numberQuestion + 1) + "/1.1");
                //дописати загрузуку чому дорівнює відповідь доробити скріни
            }
        }
    }


    public void KllikOk()
    {
        if (loader.itsTeory == true)
        {
            if (toggleGroup.AnyTogglesOn())
            {
                for (int i=0;i<tempObj.Count;i++)
                {
                    if (toggleComponents[tempObj[i]].isOn)
                    {
                        if (i ==
localizationManager.loadedData.question[numberTypeQuestion].questionDetails[numberQuestion].correctA
NS-1)
                        {
                            //NonAnsver.SetActive(false);
                            TrueAnsverTest(i);
                            loader.EndTaskCount++;
                            loader.bals++;
                            interClickOff();
                            SwitchBtn();
                        }
                        else
                        {
                            // NonAnsver.SetActive(true);
                            FalseAnsverTest(i);
                            loader.EndTaskCount++;
                            interClickOff();
                            SwitchBtn();
                        }
                    }

                }
            }
            else
            {
                Debug.Log("Оберіть відповідь");
            }

        }
```

```
        else if
(localizationManager.loadedData.question[numberTypeQuestion].questionDetailsPr[numberQuestion].regim
e == "test")
        {
            if (toggleGroup.AnyTogglesOn())
            {
                for (int i = 0; i < tempObj.Count; i++)
                {
                    if (toggleComponents[tempObj[i]].isOn)
                    {
                        if (i ==
localizationManager.loadedData.question[numberTypeQuestion].questionDetailsPr[numberQuestion].correc
tANS-1)
                        {
                            //NonAnsver.SetActive(false);
                            TrueAnsverTest(i);
                            loader.bals++;
                            loader.EndTaskCount++;
                            interClickOff();
                            SwitchBtn();
                        }
                        else
                        {
                            // NonAnsver.SetActive(true);
                            FalseAnsverTest(i);
                            loader.EndTaskCount++;
                            interClickOff();
                            SwitchBtn();
                        }
                    }
                }
            }
            else
            {
                Debug.Log("Оберіть відповідь");
            }
        }
        else
        {
            //обмеження введення
            if
(localizationManager.loadedData.question[numberTypeQuestion].questionDetailsPr[numberQuestion].corre
ctANSStr == PacticeAnver.text)
            {
                loader.bals++;
                loader.EndTaskCount++;
                Debug.Log("Відповідь вірна");
                interClickOff();
                SwitchBtn();
                TrueAnsverPr();

            }
            else
            {
                Debug.Log("Відповідь не вірна");
                interClickOff(); //времено
                SwitchBtn();//времено
                loader.EndTaskCount++; // времено
                Ansver.GetComponent<Image>().sprite= Resources.Load<Sprite>("Formulas/Practice/" +
(numberTypeQuestion + 1) + "/" + (numberQuestion + 1) + "/1");
                Ansver.SetActive(true);
                FalseAnsverPr();

                //Запуск питань підказок якшо ще раз невірно то вірна відпоідь і плюс завдання в
лоадері
            }
        }
```

```
    }
    void interClickOff()
    {
        if (loader.itsTeory == true ||
localizationManager.loadedData.question[numberTypeQuestion].questionDetailsPr[numberQuestion].regime
== "test")
        {
            foreach (Toggle n in toggleComponents)
            {
                n.GetComponent<Toggle>().interactable = false;
            }
        }
        else
        {
            EnterAnsverElement.GetComponent<InputField>().interactable = false;
        }
    }


    public void NextQuetion()
    {
        numberQuestion++;
        loader.ActiveNextQe();

    }
    void SwitchBtn()
    {
        BtnOK.SetActive(!BtnOK.activeSelf);
        BtnNextQe.SetActive(!BtnOK.activeSelf);
    }


    public void QeStar()
    {
        numberQuestion = 0;

    }
    public void LoadRezult()
    {

        loader.LoadRezult();

    }


    void TrueAnsverTest(int ansver)
    {
        toggleComponents[tempObj[ansver]].GetComponentInChildren<Image>().color = new Color(0f, 1f,
0f, 1f);
        testingImgPlase[ansver].GetComponent<QeToImage>().image.GetComponent<Image>().color = new
Color(0f, 1f, 0f, 1f);
        taskList.TrueAnsver(loader.numActiveQe);
    }
    void FalseAnsverTest(int ansver)
    {

        toggleComponents[tempObj[ansver]].GetComponentInChildren<Image>().color = new Color(1f, 0f,
0f, 1f);
        testingImgPlase[ansver].GetComponent<QeToImage>().image.GetComponent<Image>().color = new
Color(1f, 0f, 0f, 1f);


toggleComponents[tempObj[localizationManager.loadedData.question[numberTypeQuestion].questionDetails
[numberQuestion].correctANS - 1]].GetComponentInChildren<Image>().color = new Color(0f, 1f, 0f, 1f);

testingImgPlase[localizationManager.loadedData.question[numberTypeQuestion].questionDetails[numberQu
estion].correctANS - 1].GetComponent<QeToImage>().image.GetComponent<Image>().color = new Color(0f,
1f, 0f, 1f);
```

```
        taskList.FalseAnsver(loader.numActiveQe);
    }
    void FalseAnsverPr()
    {

        taskList.FalseAnsver(loader.numActiveQe);
    }
    void TrueAnsverPr()
    {

        taskList.TrueAnsver(loader.numActiveQe);
    }

    public void ButtonsOff()
    {
        Buttons.SetActive(false);
    }
}
```

## Б.14 Код скрипту «SceneManagers»

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


public class SceneManagers : MonoBehaviour
{

    public void StartSceneMenu()
    {
        SceneManager.LoadScene("MainScene");
    }
    public void StartSceneLoad()
    {
        SceneManager.LoadScene("StartScene");
    }
}
```