ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ДЕННОЇ ОСВІТИ
ФОРМА НАВЧАННЯ ДЕННА
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Допускається до захисту

Завідувач кафедри_____ Ольховська О.В.

(підпис)

«___» _____ 202_ р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОЇ РОБОТИ
НА ТЕМУ:

«Тренажер з теми «Машини Тьюрінга» англомовного дистанційного навчального курсу «Теорія алгоритмів» та розробка його програмного забезпечення»

зі спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерні науки»
ступеня магістра

Виконавець роботи Порожняков Володимир Олександрович

_____ «___» _____ 202_ р.

(підпис)

Науковий Керівник к. ф.-м. н., Черненко Оксана Олексіївна

_____ «___» _____ 202_ р.

(підпис)

Полтава – 2023

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ «ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

ЗАТВЕРДЖУЮ
Завідувач Кафедри _____Ольховська О.В.
(підпис)
«____»_____202 р.

ЗАВДАННЯ І КАЛЕНДАРНИЙ ГРАФІК ВИКОНАННЯ ДИПЛОМНОЇ РОБОТИ

Здобувач вищої освіти зі спеціальності 122 Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»
Прізвище, ім'я, по батькові Порожняков Володимир Олександрович
Тема «Тренажер з теми «Машини Тьюрінга» англомовного навчального курсу «Теорія Алгоритмів» та розробка його програмного забезпечення.»
затверджена наказом ректора № ___-Н від «__» _____ 202__ р.
Термін подання студентом дипломної роботи «__» _____ 202__ р.
Вихідні дані до магістерської роботи: публікації з теми, навчальні тренажери в дистанційних курсах із комп'ютерних наук.

Зміст пояснювальної записки:
ВСТУП.
1. ПОСТАНОВАКА ЗАДАЧІ.
2. ІНФОРМАЦІЙНИЙ ОГЛЯД.
3. ТЕОРЕТИЧНА ЧАСТИНА.
3.1. Загальні відомості.
3.2. Алгоритм програми.
3.3 Схема алгоритму.
4. ПРАКТИЧНА ЧАСТИНА.
4.1. Огляд програми.
4.2. Опис процесу створення програми.
4.3 Тестування програми.
ВИСНОВКИ.
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.
ДОДАТКИ.

Перелік графічного матеріалу: 3-4 аркуші блок-схем, інші необхідні ілюстрації.
Консультанти розділів магістерської роботи

Розділ	Призвіще, ініціали, посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1. ПОСТАНОВАКА ЗАДАЧІ.	Черненко О.О.		
2. ІНФОРМАЦІЙНИЙ ОГЛЯД.	Черненко О.О.		
3. ТЕОРЕТИЧНА ЧАСТИНА.	Черненко О.О.		
4. ПРАКТИЧНА ЧАСТИНА.	Черненко О.О.		

Календарний графік виконання роботи

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

Дата видачі завдання «__»_____202 р.

Здобувач вищої освіти Порожняков Володимир Олександрович

Науковий керівник к. ф.-м. н., Черненко О.О.

Результати захисту дипломної роботи

Дипломна робота оцінена на

_____

(балів, оцінка за національною шкалою, оцінка за ECTS)

Протокол засідання ЕК №___ від «____»_____202 р.

Секретар ЕК_____ _____

(підпис) (ініціали та призвіще)

Затверджую **Погоджено**

Зав. кафедрою_____ Науковий керівник_____

к.ф.-м.н. О. Ольховська
«____»_____202 р.

доцент, к.ф.-м.н. О. Черненко
«___»_____202 р.

План

Дипломної роботи здобувача вищої освіти ступеня магістра
спеціальності 122 Комп'ютерні науки
освітня програма 122 Комп'ютерні науки
Призвище, ім'я, по батькові Порожняков Володимир Олександрович
на тему « Тренажер з теми «Машини Тьюрінга» англомовного навчального курсу «Теорія Алгоритмів» та розробка його програмного забезпечення»

Здобувач вищої освіти_____ В. Порожняков
«___»_____202 р.

РЕФЕРАТ

Записка: 41 с., 14 рис., 1 додаток, 21 джерел.

Об’єкт розробки – Тренажер з теми «Машини Тьюрінга» англомовного навчального курсу «Теорія Алгоритмів»

Мета роботи – проектування і програмна реалізація елементів тренажера з теми «Машини Тьюрінга» англомовного навчального курсу «Теорія Алгоритмів» розробка його програмного забезпечення.»

Методи дослідження – Програмне забезпечення Було розроблено за допомогою MS Visual Studio 2022 при використані мови програмування C#

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

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

Ключові слова: ТРЕНАЖЕР, МАШИНИ ТЬЮРІНГА, ТЕОРІЯ АЛГОРИТМІВ.

ЗМІСТ

ВСТУП

Актуальність. у сучасному світі навчальний процес може натрапити на різні проблеми які у свою чергу можуть позбавити студентів можливості повноцінного аудиторного навчання, тому на зміну фізичному навчанню приходить комп'ютерні технології за допомогою яких можливо проводити навчальний процес через різноманітні програми або інтернет ресурси, плюси використання такого методу очевидні а саме, можливість навчатися де завгодно та при будь-якому часі, вимагаючи лише доступ до персонального комп'ютера.

Мета роботи – Програмування тренажеру з теми «Машини Тьюрінга»

Об'єкт розробки – програмне забезпечення для самостійного навчання студентів з теми «Машини Тьюрінга» та «Теорія алгоритмів».

Предмет роботи – предметом даної магістерської роботи є програмавчитель для навчання чи удосконалення знань з теми «Машини Тьюрінга».

Методи роботи – У ході реалізації тренажера було використане Програмне забезпечення MS Visual Studio 2022 та мови програмування C#

Тренажер може бути використаний для навчання як у лекційному форматі так і для дистанційного курсу.

Складові пояснювальної записки магістерської роботи:

- Титульний аркуш;
- Зміст;
- Вступ;
- Теоретична частина та практична частина роботи;
- Висновки;
- Список використаних джерел;
- Додатки;

1. ПОСТАНОВКА ЗАДАЧІ

Головною задачею курсової роботи є розробка навчального тренажера з теми «Машини Тьюрінга» англомовного навчального курсу «Теорія Алгоритмів».

Тренажер було створено у вигляді окремої програми, яку можливо завантажити до особистого комп‘ютера студента через дистанційний курс або інші джерела, для роботи тренажера не потрібно мати доступу до мережі інтернет.

Тренажер створений для самостійної роботи студента при вивчені відповідної теми «Теорія Алгоритмів». Тому до тренажера є певні вимоги.

Тренажер має мати прости і інтуїтивно понятий інтерфейс користувача, без складних та комплексних довідкових матеріалів.

Тренажер має бути доступним у використані для тих студентів які погано або взагалі не опанували матеріал з відповідної теми, тренажер має рахувати правильні відповіді і вказувати на помилки, щоб допомогти студенту у навчанні.

Технічні вимоги до тренажера вимагають щоб тренажер міг працювати без будь-яких проблем на різних конфігураціях персональних комп’ютерів а також різних конфігураціях операційних систем.

- Виконання поставленої задачі буде здійснюватися в наступних етапах.
- Аналіз існуючих тренажерів схожої тематики.
- Створення завдань для тренажеру.
- Розробка алгоритму роботи тренажеру.
- Розробка блок-схеми тренажеру.
- Розробка тренажеру за допомогою мов програмування.
- Тестування тренажеру для без перебійної роботи.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Огляд програмного забезпечення

За часи існування Полтавського Університету Економіки та Торгівлі було створено безліч тренажерів з різних дисциплін [3-20].

Тренажер по темі «Мови і граматики» [3] має три основні завдання, для розв’язання яких потрібно пройти декілька кроків. Кожен із кроків є запитанням із можливими варіантами відповіді. Після правильної відповіді відбувається перехід до наступного кроку. Якщо дано неправильну відповідь, то виводиться повідомлення, що містить підказку. Після цього також відбувається перехід до наступного кроку.

Також є можливість ознайомлення із теоретичним матеріалом теми. Тренажер створений за допомогою мови програмування Python.

Особливості даного тренажеру у зрозумілому і простому інтерфейсі, розбиття тесту на декілька кроків а також наявність підказок при помилках.

Тренажер [4] відноситься до теми «Алгоритмічно нерозв’язні проблеми» з дисципліни «Теорія програмування».

Уданому тренажері розглядаються такі задачі.

Проблема еквівалентності алгоритмів.

Розподіл дев’яток у запису числа π.

Обчислення досконалих чисел.

Позиціонування машини поста на останньому ящику.

Тренажер був створений за допомогою мови програмування Java. Вікно тренажера зображене на рис. 2.1.

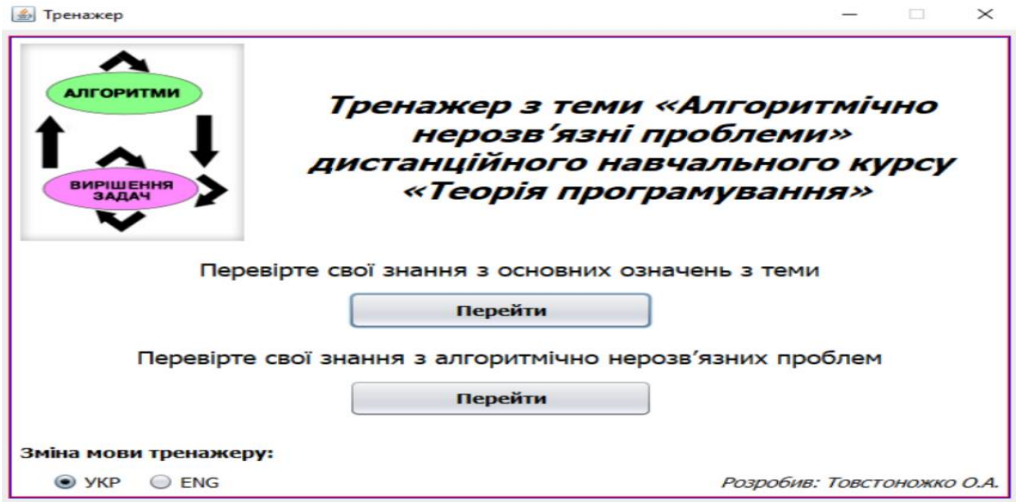


Рис.2.1 - вікно тренажера

Тренажер є набором кроків, де користувач на кожному з кроків отримує запитання в формі тесту. Користувач має обрати відповідь або вірну послідовність. Якщо було дано невірну відповідь, то користувачеві дають можливість використати вкладку із пояснювальною запискою для розуміння питання.

Серед плюсів даного тренажеру можна виділити:

Зручний та і зрозумілий графічний інтерфейс;

Наявність підказок до запитання які детально пояснюють суть питання;

Наявність двох мов як англійської так і української.

Серед недоліків можна виділити:

Невелику кількість запитань для складної теми.

В [5] маємо алгоритмізацію та програмування тренажера з теми «Способи задання мов». Тренажер а точніше його робота розбита по крокам. На кожному кроці тренажера виводиться питання та відповіді. Користувач вибирає правильну відповідь. Якщо вибрана неправильна, то виводиться повідомлення про це і користувач повторює спробу.

Тренажер створений за допомогою мови програмування C#, для роботи з інформацією використовується технологія HTML.

Серед недоліків тренажеру можна виділити те, якщо користувач помилився, він тільки отримує повідомлення про помилку, але не отримує підказки чи пояснення, чому саме відповідь неправильна.

В [10-11] тренажер з теми «Дерева розбору». Цей тренажер створений за допомогою мови програмування Java.

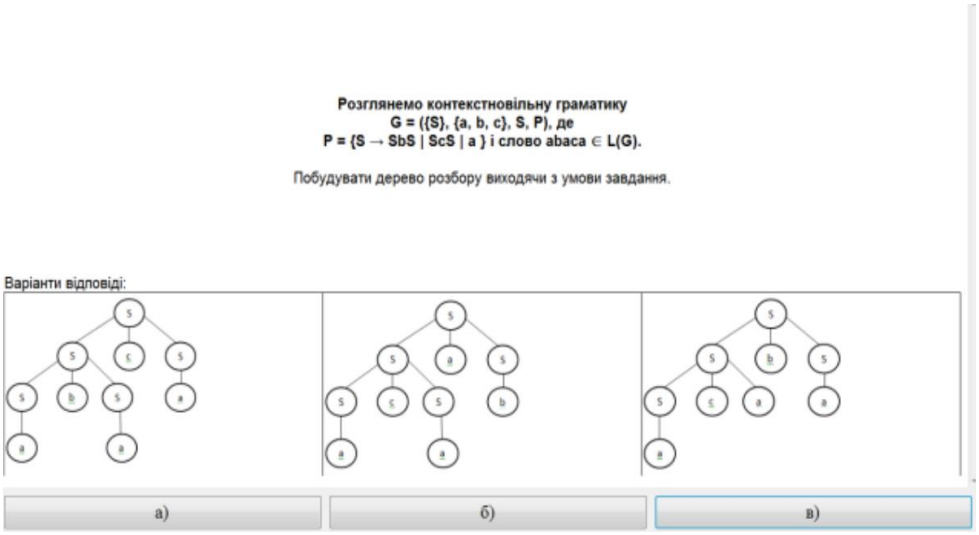


Рис. 2.2 – вікно тренажера

Робота тренажера реалізована у вигляді кроків, на рис. 1.2. маємо приклад кроку, як і завжди користувач обирає відповідь щоб перейти до наступного питання.

Серед плюсів даного тренажеру можна виділити:

Велика кількість кроків та різних запитань;

Різні рівні складності;

В [8] маємо тренажер з теми «Математичні основи теорії алгоритмів». Тренажер містить теоретичні відомості та завдання у вигляді тестів. Для створення було використано мову програмування Java.

В [6] маємо тренажер по темі «Обрізання основ», створений за допомогою мови програмування Java. Тренажер має як практичну так і теоретичну частину. Тренажер дає можливість розв’язати дві задачі. Розв’язання кожної задачі розбите на кроки. Користувачу потрібно обрати правильну відповідь на кожному кроці. У випадку коли відповідь правильна, то відбувається перехід до

наступного кроку (на рис. 2.2. показаний вигляд вікна тренажера на одному із кроків). У випадку неправильної відповіді користувач отримує змогу дати відповідь знову. Якщо користувач ще раз робить помилку, то виводиться підказка.

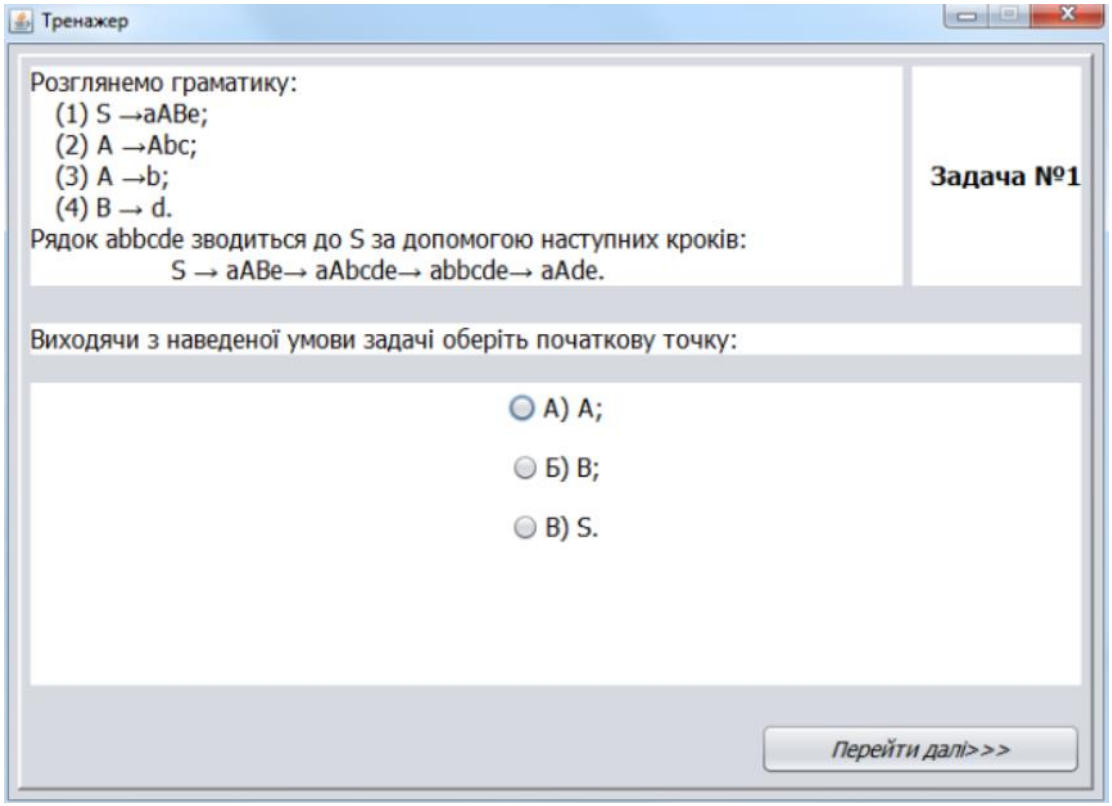


Рис. 2.3 – вікно тренажера

Після проходження всіх завдань користувач отримує інформацію про сумарну кількість помилок та пропозицію пройти тестування ще один раз.

Серед плюсів можна виділити:

наявність теоретичної інформації;

інформація про кількість помилок, яка виводиться після проходження тестування.

Серед мінусів можна виділити, відсутність пояснення у разі помилки.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1 Загальні відомості

Маши́на Тью́рінга — математичне поняття, введене для формального уточнення інтуїтивного поняття алгоритму. Названа на честь англійського математика Алана Тьюрінга, який запропонував це поняття у 1936.

Аналогічну конструкцію машини згодом і незалежно від Тьюрінга ввів американський математик Еміль Пост.

Основна ідея, що лежить в основі машини Тьюрінга, дуже проста. Машина Тьюрінга — це абстрактна машина (автомат), що працює зі стрічкою, що складається із окремих комірок, в яких записано символи.

Принцип роботи машини Тюрінга базується на ідеї виконання послідовної обробки символів на безкінечній стрічці. Вона має скінчену набір станів та символів, а також правила переходу, які визначають, як машина реагує на символи на стрічці. Машину Тюрінга можна розглядати як універсальний комп'ютер, здатний моделювати будь-який алгоритм, якщо вона отримує відповідну програму.

Машина також має голівку для запису та читання символів із комірок і яка може рухатись вздовж стрічки. На кожному кроці машина зчитує символ із комірки, на яку вказує голівка та, на основі зчитаного символу та внутрішнього стану, робиться наступний крок. При цьому, машина може змінити свій стан, записати інший символ у комірку або пересунути голівку на одну комірку ліворуч або праворуч.

У кожної машини Тьюрінга є стрічка, потенційно нескінченна в обидві сторони. Є скінченна множина символів стрічки S0, …, Sn, що

називається алфавітом машини. У кожен момент часу кожна комірка може бути зайнята не більш ніж одним символом. Машина має деяку скінченну множину внутрішніх станів q0, q1, …, qn. У кожен даний момент часу машина знаходиться лише в одному із цих станів.

Нарешті, є голівка, яка у кожен даний момент часу знаходиться на одній із комірок стрічки. Машина діє не безупинно, а лише у дискретні моменти часу. Якщо у якийсь момент t голівка сприймає комірку (тобто знаходиться на комірці), що містить символ Si, і машина знаходиться у внутрішньому стані q_j, то дія машини визначена однією із чотирьох дій:

голівка затирає символ S_i, і записує у тій же комірці новий символ S_k,

голівка пересувається в сусідню ліву комірку,

голівка пересувається в сусідню праву комірку,

машина зупиняється.

У випадках (1)-(3) машина переходить у новий внутрішній стан q_r, і готова знову до дії у наступний момент $t + 1$. Припустимо, що символ S_0 представляє порожню комірку, і отже, голівка завжди сприймає деякий символ.

Незважаючи на свій простий устрій, машина Тьюрінга може виконувати складні перетворення слів. Спочатку, коли стрічка містить вхідне слово, автомат знаходиться проти деякої з комірок і у якомусь стані. У залежності від вибору початкової комірки, утворяться різні результати роботи машини Тьюрінга. У процесі своєї роботи машина Тьюрінга буде ніби перестрибувати з однієї комірки програми на іншу, відповідно до інформації на стрічці і вказівками в командах програми, поки не дійде до команди, яка переведе автомат у *кінцевий стан* [21].

3.2 Алгоритм програми

Після запуску перед користувачем з’явиться вікно програми, у вікні програми користувач має вибір із 3 тестів, також користувач може перейти до

англійської варіації тестів, для тестування програми було додано 2 тести, один із них для тестування роботи з тестами, другий для перевірки у разі відсутності тесту.

У нашому випадку користувач обирає перший тест, після чого буде відкритий тест Тьюрінга який складається із 10 питань та варіантів відповідей до них, нижче буде представлено питання та відповіді до них:

1/10. Хто був Алан Тюрінг?

Математик.

Історик.

Фізик.

Відповідь перший варіант.

2/10. Що таке Машина Тюрінга?

Теоретична модель обчислення, яка складається з обмеженої пам'яті та правил переходу між станами.

Фізичний пристрій для виконання обчислень.

Фізико-математичний закон виконання послідовних дій.

Відповідь перший варіант.

3/10. Які основні компоненти Машини Тюрінга?

Поршневий двигун, гідравлічний кабель, камера запису.

Клавіатура, монітор, процесор.

Стрічка, головка читання/запису, стани та таблиця переходів.

Відповідь третій варіант.

4/10. Які типи рухів може здійснювати Машина Тюрінга?

Відскок, поворот.

Переміщення по площі.

Зсув вліво, зсув вправо або залишення на місці.

Відповідь третій варіант.

5/10. Як Машина Тюрінга використовує символи на своїх стрічках?

Машина Тюрінга може читати та записувати символи на стрічці, що допомагає виконувати обчислення.

Машина Тюрінга використовує стрічку як блок пам'яті.

Машина Тюрінга не використовує символи на стрічці.

Відповідь перший варіант.

6/10. Що таке стан Машини Тюрінга?

Стан Машини Тюрінга визначає поточний внутрішній стан пристрою під час виконання обчислень.

Стан Машини Тюрінга відображає кількість кроків, які вона виконала.

Стан Машини Тюрінга це коли машина тюрінга знаходиться у робочому стані.

Відповідь перший варіант.

7/10. Які основні етапи виконання програми Машини Тюрінга?

Введення вхідних даних, виконання обчислень, виведення результату.

Читання символу, застосування правила переходу, запис символу та зміна стану.

Створення правила ходу, запис скрипта, запуск стрічки.

Відповідь другий варіант.

8/10. Як Машина Тюрінга читає і записує символи на стрічці?

Машина Тюрінга використовує клавіші на клавіатурі для введення символів на стрічці.

Машина Тюрінга використовує головку читання/запису для зчитування та запису символів на стрічці.

Машина Тюрінга використовує камеру для розпізнавання матеріалу і прес для його запису.

Відповідь другий варіант.

9/10. Що таке перехід Машини Тюрінга?

Перехід Машини Тюрінга включає виконання математичних операцій над символами.

Перехід Машини Тюрінга включає зміну стану, запис символу на стрічку та зсув головки.

Перехід Машини Тюрінга включає перелік отриманої інформації.

Відповідь другий варіант.

10/10. Як Машина Тюрінга приймає або відхиляє вхідні дані?

Машина Тюрінга приймає вхідні дані якщо вони підходять для обробки і відхиляє якщо не підходять.

Машина Тюрінга приймає вхідні дані, якщо вона зупиняється у стані прийняття, а відхиляє, якщо зупиняється у стані відхилення.

Машина Тюрінга приймає або відхиляє вхідні дані, залежно від значень символів на стрічці.

Відповідь перший варіант.

Після проходження тесту користувач отримає оцінку за 5 бальною шкалою, також буде отримана інформація де буде вказано питання на які було дано не вірну відповідь, після проходження тесту користувач може розпочати його знову, обрати інший тест або вийти з програми, переходимо до другого тесту, нижче буде представлено питання та відповіді до них:

1/10. Які завдання можна вирішувати за допомогою Машини Тюрінга?

Машини Тюрінга можуть вирішувати різноманітні обчислювальні завдання, включаючи обробку мов, розрахунки та симуляції.

Машини Тюрінга призначені лише для виконання арифметичних операцій.

Машини Тюрінга можуть вирішувати лише фізичні проблеми.

Відповідь перший варіант.

2/10. Що таке універсальна Машина Тюрінга?

Універсальна Машина Тюрінга - це Машина Тюрінга, яка може симулювати будь-яку іншу Машину Тюрінга.

Універсальна Машина Тюрінга - це Машина Тюрінга, яка може вирішувати будь-яку обчислювальну задачу.

Універсальна Машина Тюрінга - це Машина Тюрінга, яка може створювати будь-яку фізичну модель.

Відповідь перший варіант.

3/10. Які обмеження існують у моделі Машини Тюрінга?

Модель Машини Тюрінга має обмеження в обсязі пам'яті та часу виконання.

Модель Машини Тюрінга не має обмежень у розмірі пам'яті та часу виконання.

Модель Машини Тюрінга має обмеження у розмірах та вигляді.

Відповідь перший варіант.

4/10. Як Машина Тюрінга використовує пам'ять?

Машина Тюрінга використовує стрічку як свою пам'ять, де символи можуть бути читані, записувані та модифіковані головкою.

Машина Тюрінга використовує оперативну пам'ять та жорсткий диск для зберігання даних.

Машина Тюрінга використовує блок-пам'яті у вигляді касет.

Відповідь перший варіант.

5/10. Як Машина Тюрінга може розв'язувати проблеми шифрування?

Машина Тюрінга може моделювати алгоритми шифрування та розшифрування для аналізу та розв'язання криптографічних проблем.

Машина Тюрінга не може вирішувати проблеми шифрування, це виключно завдання криптоаналітиків.

Машина Тюрінга може розв'язувати проблеми шифрування але у обмеженому виді.

Відповідь перший варіант.

6/10. Які основні принципи Тези Черча-Тюрінга?

Тези Черча-Тюрінга стверджують, що будь-яка обчислювальна задача, яку можна алгоритмічно вирішити, може бути розв'язана Машиною Тюрінга.

Тези Черча-Тюрінга обмежують можливості Машини Тюрінга, стверджуючи, що вона не може розв'язувати певні класи обчислювальних задач.

Тези Черча-Тюрінга стверджують, що будь-яка обчислювальна задача може бути розібрана на основні частини із яких вона складається через Машину Тюрінга.

Відповідь перший варіант.

7/10. Які розширення Машини Тюрінга були запропоновані для моделювання обчислень?

Розширення Машини Тюрінга включають недетермінізм, багатолінточність, оракули та інші.

Машини Тюрінга не можуть бути розширені для моделювання інших видів обчислень.

Машини Тюрінга можуть бути розширені через збільшення розміру самої машини.

Відповідь перший варіант.

8/10. Які відомі проблеми нерозв'язані для Машини Тюрінга?

Деякі проблеми, які не мають алгоритмічного розв'язку, відомі як нерозв'язувані проблеми Машини Тюрінга, наприклад, проблема зупинки.

Машини Тюрінга можуть розв'язати будь-яку обчислювальну проблему.

Проблеми Фізико-хімічні проблеми не можуть бути розв'язаними через Машину Тюрінга, тільки Математичні.

Відповідь перший варіант.

9/10. Що таке недетерміністична Машина Тюрінга?

Недетерміністична Машина Тюрінга - це Машина Тюрінга, у якої існує можливість зробити кілька можливих переходів з одного стану до іншого.

Недетерміністична Машина Тюрінга - це Машина Тюрінга, яка виконує обчислення у випадковому порядку.

Недетерміністична Машина Тюрінга - це Машина Тюрінга, яка вирішує однакові проблеми але отримує різні відповіді.

Відповідь перший варіант.

10/10. Чому Машину Тюрінга вважають універсальним обчислювальним пристроєм?

Машина Тюрінга вважається універсальною, оскільки вона може симулювати будь-яку іншу Машину Тюрінга або обчислювальний процес.

Машина Тюрінга вважається універсальною через свою надзвичайно високу швидкість обчислень.

Машина Тюрінга вважається універсальною через можливість вирішити майже усі проблеми.

Після проходження тесту користувач як і у першому випадку проходження тесту отримає оцінку та інформацію про помилки, переходимо до третього тесту, нижче буде представлено питання та відповіді до них:

1/10. Які основні компоненти складають Машину Тюрінга?

Основні компоненти Машини Тюрінга включають стрічку, головку читання/запису, стани та правила переходу.

Машини Тюрінга не мають окремих компонентів, вони працюють в цілісності.

Машина Тюрінга має основний компонент у вигляді блоку пам'яті.

Відповідь перший варіант.

2/10. Що таке детермінована Машина Тюрінга?

Детермінована Машина Тюрінга - це Машина Тюрінга, у якої для кожного стану та символу на стрічці існує єдиний можливий перехід.

Детермінована Машина Тюрінга - це Машина Тюрінга, у якої обчислення відбуваються за випадковими правилами.

Детермінована Машина Тюрінга - це Машина Тюрінга, у якої кількість символів не може бути безкінечним.

Відповідь перший варіант.

3/10. Що таке універсальна обчислювальна мова?

Універсальна обчислювальна мова - це мова програмування або формальна система, у якій можна виразити будь-яку обчислювальну функцію.

Універсальна обчислювальна мова - це мова, яку розуміють всі Машини Тюрінга.

Універсальна обчислювальна мова - це мова програмування яка використовує 4 основних компонента для відтворення будь яких функцій.

Відповідь перший варіант.

4/10. Які основні класи складності проблем визначені для Машини Тюрінга?

Основні класи складності проблем для Машини Тюрінга включають P, NP, NP-повність та NP-важкість.

Машини Тюрінга не розрізняють класи складності проблем.

Машини Тюрінга розподіляють класи проблем такі як R, T, TP.

Відповідь перший варіант.

5/10. Чому Машина Тюрінга є теоретичним концептом, а не конкретною фізичною реалізацією?

Машина Тюрінга є теоретичним концептом, оскільки вона слугує моделлю обчислень і може бути реалізована різними фізичними системами.

Машина Тюрінга не може бути конкретною фізичною реалізацією через свої обмеження в обсязі пам'яті та часу.

Машина Тюрінга не може бути фізичною концепцією через безкінечність роботи.

Відповідь перший варіант.

6/10. Чи може Машина Тюрінга мати модифікації?

Машина Тюрінга може мати будь-які модифікації у рамках концепції Машини Тюрінга.

Машина Тюрінга не може мати модифікації.

Машина Тюрінга може бути модифікована будь яким чином.

Відповідь перший варіант.

7/10. Як Машина Тюрінга виконує кроки обчислень?

Машина Тюрінга виконує обчислення шляхом зчитування символу зі стрічки, застосування правила переходу для зміни стану та запису символу на стрічку.

Машина Тюрінга виконує обчислення шляхом випадкових переходів між станами та символами на стрічці.

Машина Тюрінга виконує обчислення шляхом обчислення пам'яті у будь якому порядку.

Відповідь перший варіант.

8/10. Що таке Багатолінточна Машина Тюрінга?

Багатолінточна Машина Тюрінга - це розширення Машини Тюрінга, у якій є кілька незалежних стрічок та головок для читання/запису на цих стрічках.

Багатолінточна Машина Тюрінга - це Машина Тюрінга, яка виконує обчислення швидше за звичайну однолінточну Машину Тюрінга.

Многолінточна Машина Тюрінга - це розширення Машини Тюрінга, для вирішення будь-яких проблем.

Відповідь перший варіант.

9/10. Як Машина Тюрінга виконує кроки обчислень?

Машина Тюрінга виконує обчислення шляхом зчитування символу зі стрічки, застосування правила переходу для зміни стану та запису символу на стрічку.

Машина Тюрінга виконує обчислення шляхом випадкових переходів між станами та символами на стрічці.

Машина Тюрінга виконує обчислення шляхом зчитування символу зі стрічки у різному порядку.

Відповідь перший варіант.

10/10. Які обмеження властиві Машині Тюрінга?

Машини Тюрінга мають обмеження в обсязі пам'яті, швидкості обчислень та доступу до невизначених функцій.

Машини Тюрінга не мають жодних обмежень, вони можуть вирішити будь-яку обчислювальну задачу.

Машини Тюрінга можуть мати проблеми в залежності від різновида Машини Тюрінга

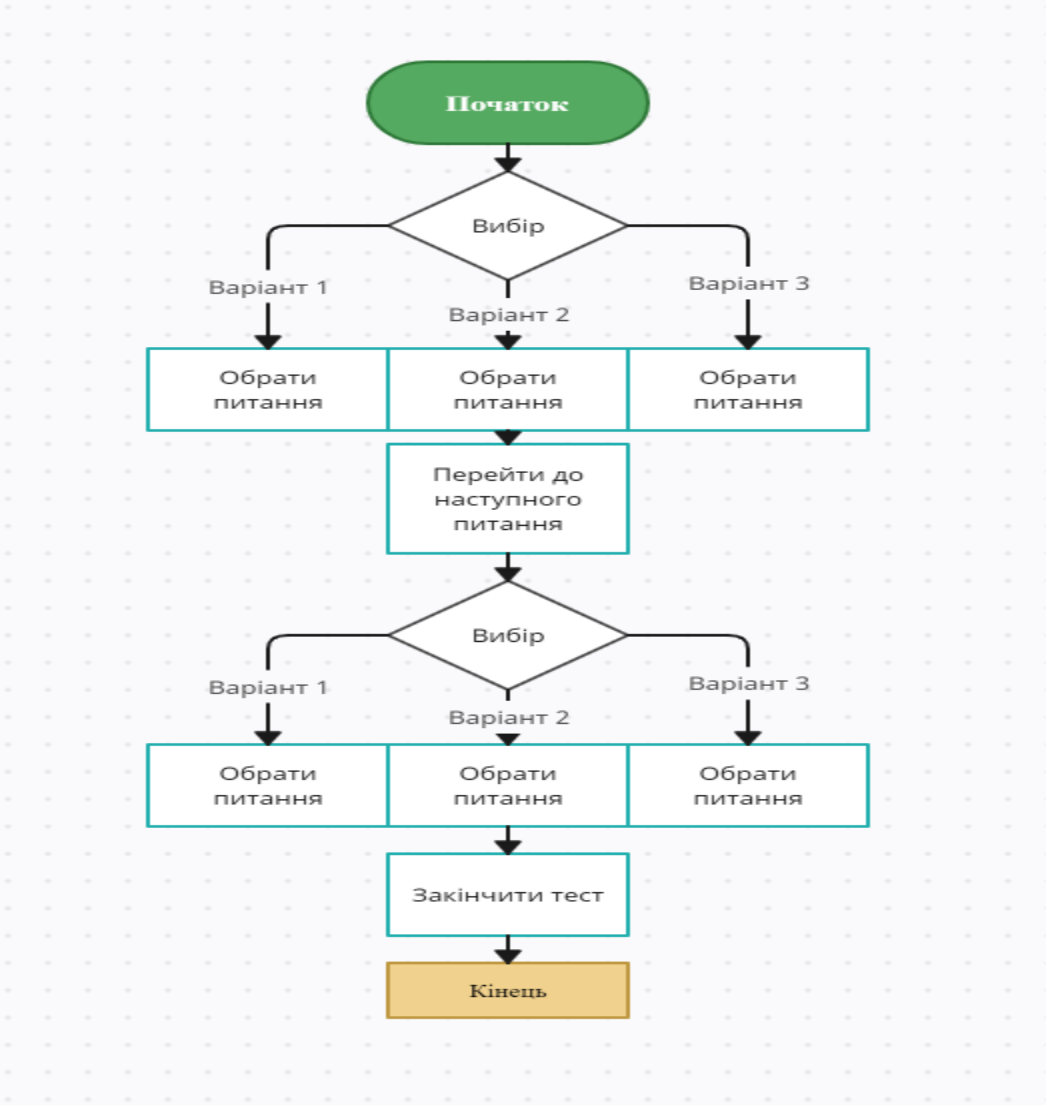
Відповідь перший варіант.

Після проходження тесту як і в першому випадку користувач отримає оцінку та інформацію про помилки, також користувач може пройти англійську варіацію тестів, для чого треба обрати кнопку «Тести 1-3 англ.» - це дубльовані тести як у разі з першими 3 тестами але переведені на англійську мову, останні два тести не призначенні для виконання, вони створенні для тестування програми.

2.2 Схема алгоритму

Після початку, користувач має вибір з трьох варіантів відповідей на вибір, користувач обирає відповідь після чого у нього з'являється можливість перейти до наступного питання, питання на яке відповів користувач буде збережене у пам’яті програми до кінця тесту, після проходження всіх тестів користувач отримує оцінку кількість правильних відповідей а також окреме вікно де буде вказано питання на які було дано не правильні відповіді.

Для візуального розуміння процесу було створено блок-схему алгоритму (Рис 3.1.1)



Блоксхема 3.1.1 – алгоритм проходження тесту.

4. ПРАКТИЧНА ЧАСТИНА

4.1 Огляд програмного продукту

Робота тренажеру у відображена рисунками

Після запуску програми ми попадемо на головний екран де можемо вибрати тест для проходження (Рис 4.1.1)

Рис 4.1.1 – Огляд програми після запуску.

Вибравши тест з’явиться питання тесту а також варіанти відповідей (Рис 4.1.1)

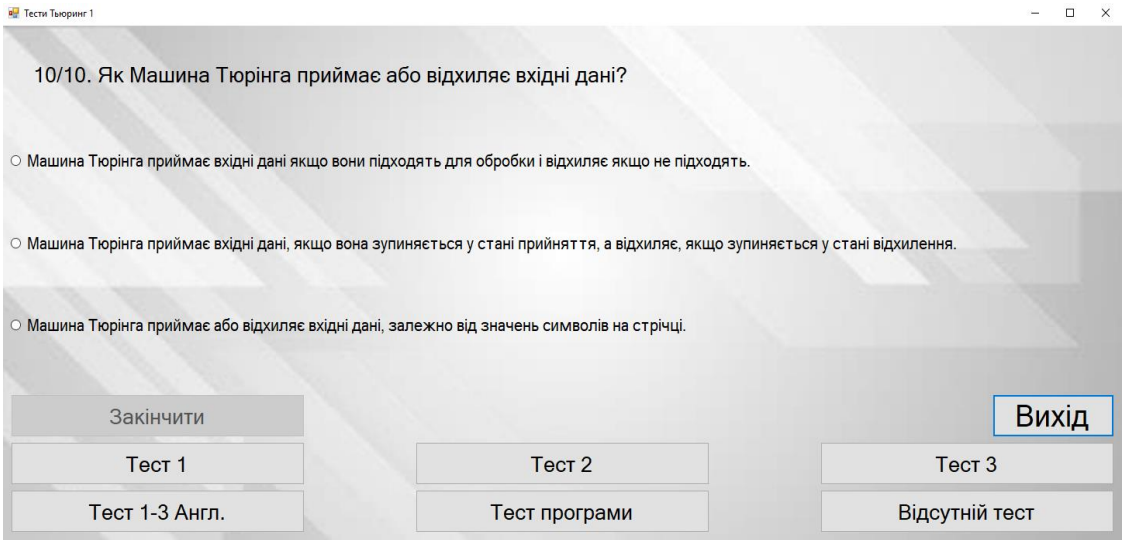


Рисунок 4.1.2 – Огляд програми при проходженні першого тесту.

Коли тест буде закінчено користувач отримає оцінку а також інформацію про помилки (Рис 4.1.3)

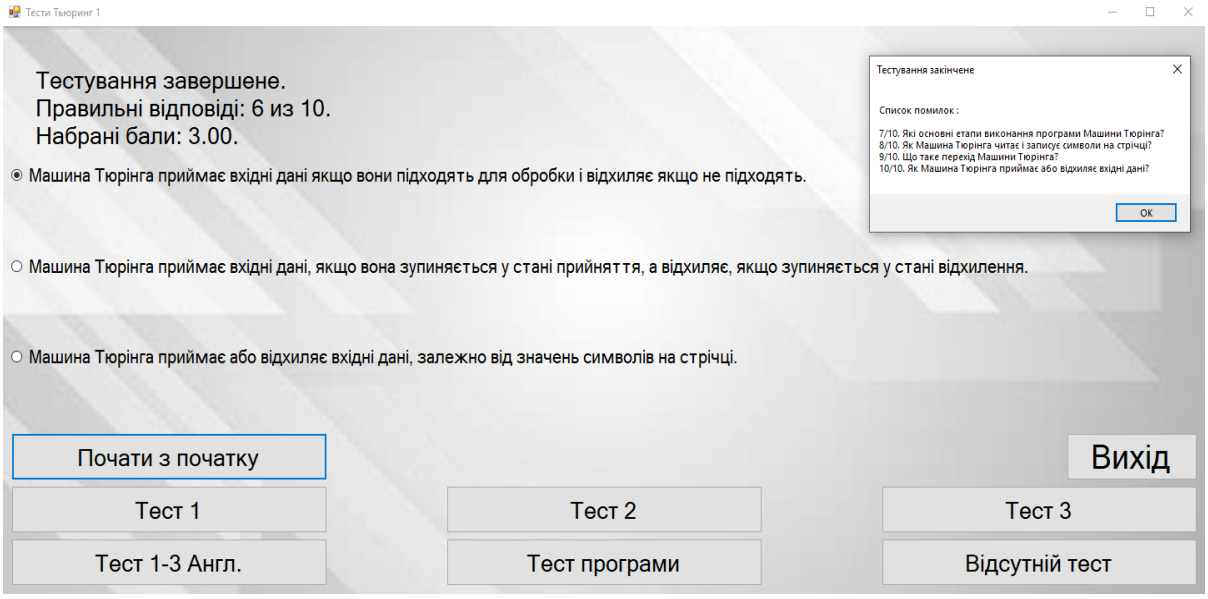


Рис 4.1.3 – Огляд програми при отримані оцінки за перший тест.

Тепер користувач може вийти з програми або обрати інший тест.

4.2 Опис процесу створення програми

Для створення програми було обрано середовище програмування Visual Studio 2022 та мову програмування C#.

Для створення вікна програми було використано середовище програмування Visual Studio 2022, першим було створення проекту типу Windows form app. (Рис.4.2.1)

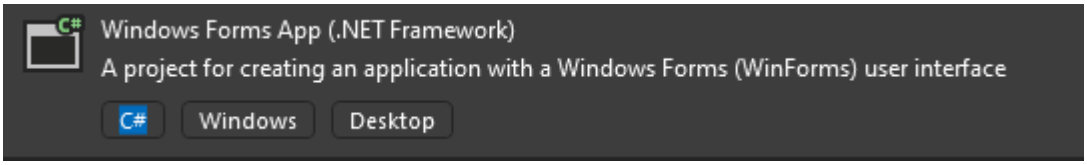


Рисунок 4.2.1 – середовищ програмування Visual Studio 2022

Тепер коли ми маємо вікно майбутньої програми на треба розмістити усі необхідні елементи, нам буде потрібно перейти до вкладки інструментів де буде взято 8 кнопок(Button), 3 перемикача(RadioButton), 1 рамка для тексту(Label). Розміщуємо усі елементи на вікні майбутньої програми, після чого треба проклікати усі елементи які було додано щоб створити їх у коді проекту. На Рисунках (Рис 4.2.2) – (Рис 4.2.4) зображено процес роботи над програмою.

Рис 4.2.2 – Робота з вікном програми.

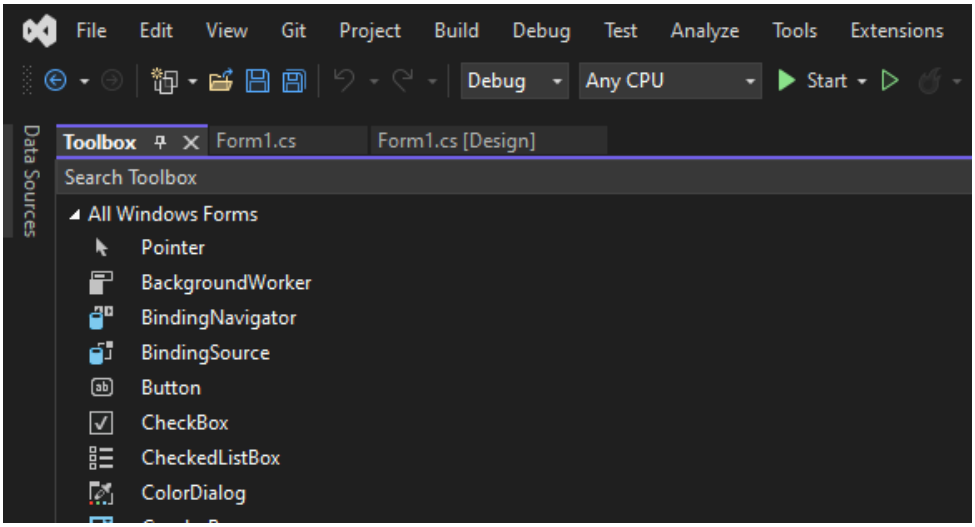


Рис 4.2.3 – Панель інструментів.

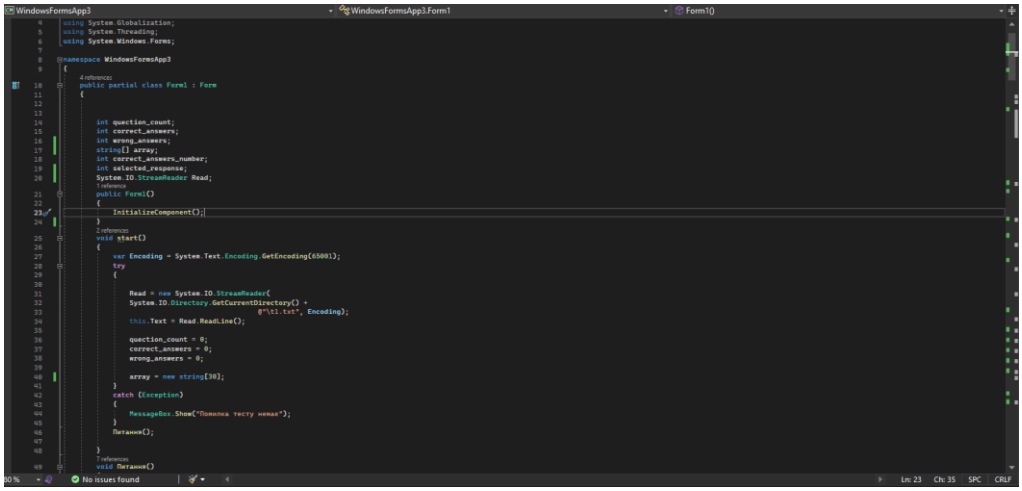


Рис 4.2.4 – Код програми.

4.3 Тестування програми

Тестування програми було проведено за допомогою декількох незалежних випробувачів, тестування було проведено таким чином:

Перевірка роботи усіх заявлених можливостей програми.

Перевірка роботи усіх кнопок програми.

Перевірка довготривалої роботи програми на безперебійну роботу.

Перевірка у разі відсутності тесту для проходження.

Перевірка оцінки за виконання тесту.

Перевірка інформації про помилки при виконані тесту.

Перевірка можливості переходити до іншого тесту у будь-який момент.

Після проходження тестування, усі рецензенти дали позитивну відповідь на по роботі із програмою, за роботою програми не було виявлено проблем.

У ході тестування рецензенти запропонували додати нові функції.

Таким чином було додано нові функції тренажеру у вигляді додаткової плашки із різним функціоналом, вигляд нового функціоналу на рисунках (4.3.1) (4.3.2) (4.3.3).

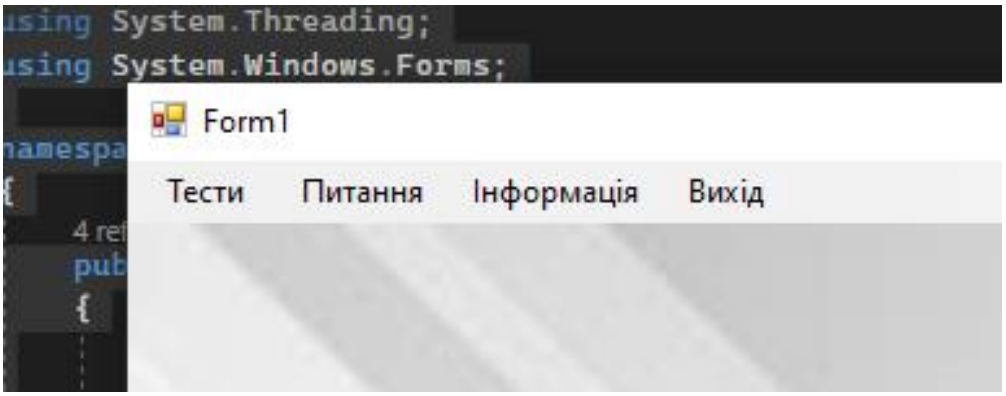


Рис. 4.3.1 – вигляд нового функціоналу

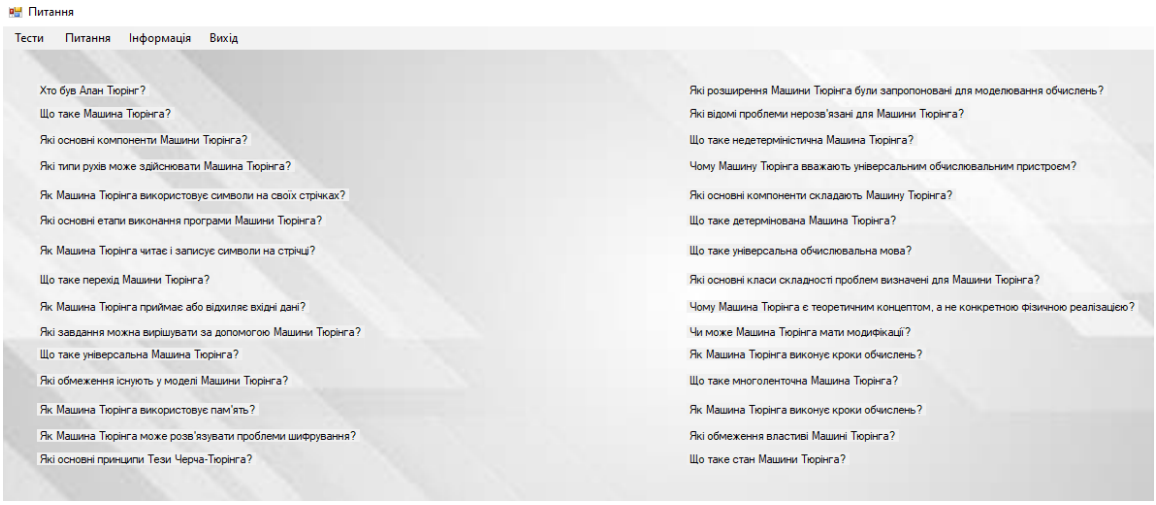


Рис. 4.3.2 – можливість переглянути усі питання тесту

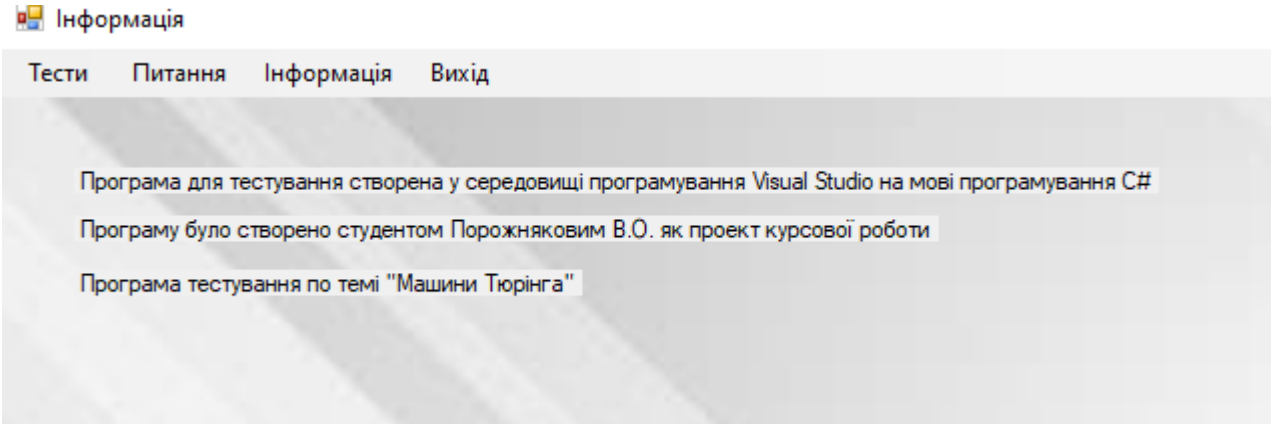


Рис. 4.3.3 – Інформаційне табло

ВИСНОВКИ

Складові магістерської роботи:

- Створено постановку задачі;
- Обрано метод розробки програми;
- Створений алгоритм роботи тренажеру;
- Програмно реалізований тренажер на мові програмування C# у середовищі програмування Visual Studio 2022;
- Реалізовано завдання для тренажеру у вигляді тестів;
- Реалізований інтуєтивно зрозумілий та зручний інтерфейс користувача;
- Реалізовано отримання оцінки та інформацію про помилки після проходження тесту;
- Можливість знову пройти тест через кнопку «почати з початку» після завершення тесту;
- Проведене тестування тренажеру;
- Можливість перегляду усіх питань наявних у тренажері;
- Тренажер може бути запущеним на будь-якому комп’ютері із ОС Windows;

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Черненко О. О. Методичні підходи щодо створення дистанційного курсу з дисципліни "Теорія програмування"/ О. О.Черненко// Інформатика та системні науки (ІСН-2017): матеріали VIII Всеукраїнської науково-практичної конференції за міжнародною участю (м. Полтава, 16–18 березня 2017 р.) – Полтава: ПУЕТ, 2017. – С. 285-286. (http://dspace.puet.edu.ua/handle/123456789/5591)

2. Теркун Д.С. Удосконалення електронного навчального посібника з дисципліни "Теорія програмування" / Д.С. Теркун // Інформатика та системні науки (ІСН2012) : матеріали III Всеукр-.наук.-практ. конф., (м. Полтава, 1–3 березня 2012 р.). – Полтава: ПУЕТ, 2012. – С. 250-251.

3. Данник О. І. Навчальний тренажер з теми «Мови і граматики» та його програмна реалізація / О. І. Данник, О. О. Черненко // Комп'ютерні науки і прикладна математика (КНіПМ-2018): матеріали науково-практичного семінару. Випуск 2 – Полтава: Кафедра ММСІ ПУЕТ, 2018. – С.4-5.

4. Товстоножко О. С. Пояснювальна записка до дипломної роботи на тему «Розробка програмного забезпечення тренажера з теми «Алгоритмічно нерозв'язні проблеми» дистанційного навчального курсу «Теорія програмування» / О. С. Товстоножко – Полтава: ПУЕТ, 2020. – 47 с. (http://dspace.puet.edu.ua/handle/123456789/10047)

5. Величко А. О. Пояснювальна записка до бакалаврської роботи на тему «Програмне забезпечення для тренажера з теми «Способи задання мов» дистанційного навчального курсу «Теорія програмування» / А. О. Величко – Полтава: ПУЕТ, 2020. – 57 с. (http://dspace.puet.edu.ua/handle/123456789/9031)

6. Цехмейстер В. О. Пояснювальна записка до бакалаврської роботи на тему «Тренажер з теми «Метод обрізання основ» дистанційного навчального курсу «Теорія програмування» та розробка його програмного забезпечення» / В. О. Цехмейстер – Полтава: ПУЕТ, 2020. – 35 с. (http://dspace.puet.edu.ua/handle/123456789/9028)

7. Скромінський М. В. Пояснювальна записка до бакалаврської роботи на тему «Розробка програмного забезпечення для тренажера з теми «Контекстовільні граматики» дистанційного навчального курсу «Теорія програмування» / М. В. Цехмейстер – Полтава: ПУЕТ, 2020. – 39 с. (http://dspace.puet.edu.ua/handle/123456789/9025)

8. Костромін І. І. Пояснювальна записка до бакалаврської роботи на тему «Тренажер з теми «Математичні основи теорії алгоритмів» дистанційного навчального курсу «Теорія програмування» та розробка його програмного забезпечення» / І. І. Костромін – Полтава: ПУЕТ, 2020. – 49 с. (http://dspace.puet.edu.ua/handle/123456789/9005)

9. Костромін І. І. Пояснювальна записка до бакалаврської роботи на тему «Тренажер з теми «Математичні основи теорії алгоритмів» дистанційного навчального курсу «Теорія програмування» та розробка його програмного забезпечення» / І. І. Костромін – Полтава: ПУЕТ, 2020. – 49 с. (http://dspace.puet.edu.ua/handle/123456789/9005)

10. Алексов С. В. Пояснювальна записка до бакалаврської роботи на тему «Алгоритмізація та програмування тренажера «Дерева розбору» дистанційного навчального курсу «Теорія програмування» / С. В. Алексов – Полтава: ПУЕТ, 2020. – 84 с. (http://dspace.puet.edu.ua/handle/123456789/8998)

11. Алексов С. В. Тренажер з теми «Дерево розбору» дистанційного навчального курсу «Теорія програмування» та розробка його програмного забезпечення / С. В. Алексов, О. О. Черненко // Актуальні питання розвитку науки та забезпечення якості освіти у XXI столітті: тези доповідей XLIII Міжнародної наукової студентської конференції за підсумками науково-

дослідних робіт студентів за 2019 рік (м. Полтава, 07–08 квітня 2020 року). Частина 2 – Полтава: ПУЕТ, 2020 – С.115-117.

12. Белінський О. Б. Пояснювальна записка до бакалаврської роботи на тему «Тренажер з теми «Побудова предикативних синтаксичних аналізаторів» дистанційного навчального курсу «Теорія програмування» та розробка його програмного забезпечення» / О. Б. Белінський – Полтава: ПУЕТ, 2020. – 48 с. (http://dspace.puet.edu.ua/handle/123456789/8999)

13. Волосевич М. В. Тренажер з теми «Марківські підстановки» дистанційного навчального курсу «Теорія програмування» та розробка його програмного забезпечення / М. В. Волосевич, О. О. Черненко // Актуальні питання розвитку науки та забезпечення якості освіти у XXI столітті: тези доповідей XLIII Міжнародної наукової студентської конференції за підсумками науководослідних робіт студентів за 2019 рік (м. Полтава, 07–08 квітня 2020 року). Частина 2 – Полтава: ПУЕТ, 2020 – С.119-121.

14. . Ємець О.О. Про розробку тренажерів для дистанційних курсів кафедрою ММСІ ПУЕТ / О.О. Ємець // 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. – С. 152-161.

15. Ольховська О. В. Технології підтримки системи дистанційного навчання в Полтавському університеті економіки і торгівлі / О. В. Ольховська, Д. М. Ольховський // Інформатика та системні науки (ІСН-2016): матеріали VII Всеукраїнської науково-практичної конференції за міжнародною участю, (м. Полтава, 10–12 берез. 2016 р.). – Полтава: ПУЕТ, 2016. – С. 219-221. (http://dspace.puet.edu.ua/handle/123456789/3074)

16. Чілікіна Т. В. Огляд тренажерів з дисципліни "Математичний аналіз" на прикладі розробок студентів напряму "Інформатика" / Т. В. Чілікіна // Інформатика та системні науки (ІСН-2016): матеріали VII Всеукраїнської науково-практичної конференції за міжнародною участю, (м. Полтава, 10–12

берез. 2016 р.). – Полтава: ПУЕТ, 2016. – С. 329-330. (http://dspace.puet.edu.ua/handle/123456789/2993)

17. Парфьонова Т. О. Про розробку тренажерів для дистанційного навчального курсу "Алгебра і геометрія" / Т. О. Парфьонова // Інформатика та системні науки (ІСН-2016): матеріали VII Всеукраїнської науково-практичної конференції за міжнародною участю, (м. Полтава, 10–12 берез. 2016 р.) – Полтава: ПУЕТ, 2016.

18. Кондрашев Д. М. Розробка тренажера дистанційного навчального курсу "Математична логіка та теорія алгоритмів" з теми "Машини Тюрінга" / 59 Д. М. Кондрашев // Інформатика та системні науки (ІСН-2017): матеріали VIII Всеукраїнської науково-практичної конференції за міжнародною участю, (м. Полтава, 16-18 березня 2017 р.) – Полтава: ПУЕТ, 2017. (http://dspace.puet.edu.ua/handle/123456789/5497)

19. Сокол О. В. Розробка тренажера з теми «Нормальні алгоритми» дистанційного навчального курсу «Теорія алгоритмів» / О. В. Сокол, О. О. Черненко //Інформатика та системні науки (ІСН-2017): матеріали VIII Всеукраїнської науково-практичної конференції за міжнародною участю, (м. Полтава, 16- 18 березня 2017 р.) – Полтава: ПУЕТ, 2017. (http://dspace.puet.edu.ua/handle/123456789/5481)

20. Потерайло О. О. Розробка програмного забезпечення тренажера з теми «Застосування апарата алгебри логіки до логіко-математичної практики» дистанційного навчального курсу «Математична логіка та теорія алгоритмів» / О. О. Потерайло // 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. (http://dspace.puet.edu.ua/handle/123456789/2503)

21. Машина Тюрінга – (uk.wikipedia.org/wiki/Машина_Тюрінга)

Додатки

Програмний код

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Globalization;
using System.Threading;
using System.Windows.Forms;

namespace WindowsFormsApp3
{
    public partial class Form1 : Form
    {


        int quection_count;
        int correct_answers;
        int wrong_answers;
        string[] array;
        int correct_answers_number;
        int selected_response;
        System.IO.StreamReader Read;
        public Form1()
        {
            InitializeComponent();
        }
        void start()
        {
            var Encoding = System.Text.Encoding.GetEncoding(65001);
            try
            {

                Read = new System.IO.StreamReader(
                System.IO.Directory.GetCurrentDirectory() +
                                        @"\t1.txt", Encoding);
                this.Text = Read.ReadLine();

                quection_count = 0;
                correct_answers = 0;
```

```
        wrong_answers = 0;

        array = new string[30];
    }
    catch (Exception)
    {
        MessageBox.Show("□□□□□□□□ □□□□□□ □□□□□□");
    }
    □□□□□□□□();

}
void □□□□□□□□()
{
    label1.Text = Read.ReadLine();

    radioButton1.Text = Read.ReadLine();
    radioButton2.Text = Read.ReadLine();
    radioButton3.Text = Read.ReadLine();


    correct_answers_number = int.Parse(Read.ReadLine());

    radioButton1.Checked = false;
    radioButton2.Checked = false;
    radioButton3.Checked = false;


    button1.Enabled = false;
    quection_count = quection_count + 1;

    label3.Visible = false;
                        =
    label32.Visible = false;


    if (Read.EndOfStream == true) button1.Text = "□□□□□□□□□□";


}
void turn(object sender, EventArgs e)
{

    button1.Enabled = true; button1.Focus();
    RadioButton □□□□□□□□□□ = (RadioButton)sender;
    var tmp = □□□□□□□□□□.Name;

    selected_response = int.Parse(tmp.Substring(11));
}
private void label1_Click(object sender, EventArgs e)
{

}
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
```

```
}
private void radioButton2_CheckedChanged(object sender, EventArgs e)
{

}
private void radioButton3_CheckedChanged(object sender, EventArgs e)
{

}
private void button1_Click(object sender, EventArgs e)
{

   if (selected_response == correct_answers_number) correct_answers =
                                correct_answers + 1;
   if (selected_response != correct_answers_number)
   {

      wrong_answers = wrong_answers + 1;

      array[wrong_answers] = label1.Text;
   }
   if (button1.Text == "□□□□□□□ □ □□□□□□□□")
   {
      button1.Text = "□□□□□□□□□ □□□□□□□□";

      radioButton1.Visible = true;
      radioButton2.Visible = true;
      radioButton3.Visible = true;

      start();
      return;

   }

   else if (button1.Text == "□□□□□□□□□□")
   {

      Read.Close();

      radioButton1.Visible = true;
      radioButton2.Visible = true;
      radioButton3.Visible = true;

      label1.Text = String.Format("□□□□□□□□□□□ □□□□□□□□□□.\n" +
         "□□□□□□□□□□ □□□□□□□□□□: {0} □□ {1}.\n" +
         "□□□□□□□□ □□□□: {2:F2}.", correct_answers,
         quection_count, (correct_answers * 5.0F) / quection_count);

      button1.Text = "□□□□□□□ □ □□□□□□□□";

      var Str = "□□□□□□□ □□□□□□□□ " +
            ":\n\n";
      for (int i = 1; i <= wrong_answers; i++)
```

```
            Str = Str + array[i] + "\n";


        if (wrong_answers != 0) MessageBox.Show(
                            Str, "□□□□□□□□□□□ □□□□□□□□□□");
    }
    else if (button1.Text == "□□□□□□□□ □□□□□□□□")
    {
        □□□□□□□□();
    }

}

private void Form1_Load(object sender, EventArgs e)
{
    button1.Text = "□□□□□□□□ □□□□□□□□";

    button3.Text = "□□□□ 1";
    button4.Text = "□□□□ 2";
    button5.Text = "□□□□ 3";
    button6.Text = "□□□□ □□□□□□□□";
    button7.Text = "□□□□ 1-3 □□□□.";
    button8.Text = "□□□□□□□□□ □□□□";

    radioButton1.Visible = false;
    radioButton2.Visible = false;
    radioButton3.Visible = false;

    button1.Visible = false;

    label3.Visible = false;
                        =
    label32.Visible = false;




    radioButton1.CheckedChanged += new EventHandler(turn);
    radioButton2.CheckedChanged += new EventHandler(turn);
    radioButton3.CheckedChanged += new EventHandler(turn);


}
void start2()
{
    var Encoding = System.Text.Encoding.GetEncoding(65001);
    try
    {

        Read = new System.IO.StreamReader(
        System.IO.Directory.GetCurrentDirectory() +
                            @"\t2.txt", Encoding);
        this.Text = Read.ReadLine();
```

```
        quection_count = 0;
        correct_answers = 0;
        wrong_answers = 0;

        array = new String[30];
    }
    catch (Exception)
    {
        MessageBox.Show("□□□□□□□□ □□□□□□ □□□□□□");
    }
    □□□□□□□□();

}
void start3()
{
    var Encoding = System.Text.Encoding.GetEncoding(65001);
    try
    {

        Read = new System.IO.StreamReader(
        System.IO.Directory.GetCurrentDirectory() +
                                @"\t3.txt", Encoding);
        this.Text = Read.ReadLine();

        quection_count = 0;
        correct_answers = 0;
        wrong_answers = 0;

        array = new String[30];
    }
    catch (Exception)
    {
        MessageBox.Show("□□□□□□□□ □□□□□□ □□□□□□");
    }
    □□□□□□□□();

}
void start4()
{
    var Encoding = System.Text.Encoding.GetEncoding(65001);
    try
    {

        Read = new System.IO.StreamReader(
        System.IO.Directory.GetCurrentDirectory() +
                                @"\t4.txt", Encoding);
        this.Text = Read.ReadLine();

        quection_count = 0;
        correct_answers = 0;
        wrong_answers = 0;

        array = new String[11];
    }
```

```
    catch (Exception)
    {
        MessageBox.Show("□□□□□□□□ □□□□□□ □□□□□□");
    }
    □□□□□□□□();

}
void start5()
{
    var Encoding = System.Text.Encoding.GetEncoding(65001);
    try
    {

        Read = new System.IO.StreamReader(
        System.IO.Directory.GetCurrentDirectory() +
                                @"\t5.txt", Encoding);
        this.Text = Read.ReadLine();

        quection_count = 0;
        correct_answers = 0;
        wrong_answers = 0;

        array = new String[30];
    }
    catch (Exception)
    {
        MessageBox.Show("No test Found");
    }
    □□□□□□□□();

}
void start7()
{
    var Encoding = System.Text.Encoding.GetEncoding(65001);
    try
    {

        Read = new System.IO.StreamReader(
        System.IO.Directory.GetCurrentDirectory() +
                                @"\t7.txt", Encoding);
        this.Text = Read.ReadLine();

        quection_count = 0;
        correct_answers = 0;
        wrong_answers = 0;

        array = new String[30];
    }
    catch (Exception)
    {
        MessageBox.Show("No test Found");
    }
    □□□□□□□□();
```

```
}
void start8()
{
    var Encoding = System.Text.Encoding.GetEncoding(65001);
    try
    {

        Read = new System.IO.StreamReader(
        System.IO.Directory.GetCurrentDirectory() +
                                @"\t8.txt", Encoding);
        this.Text = Read.ReadLine();

        quection_count = 0;
        correct_answers = 0;
        wrong_answers = 0;

        array = new String[30];
    }
    catch (Exception)
    {
        MessageBox.Show("No test Found");
    }
    □□□□□□□□();

}
void start9()
{
    var Encoding = System.Text.Encoding.GetEncoding(65001);
    try
    {

        Read = new System.IO.StreamReader(
        System.IO.Directory.GetCurrentDirectory() +
                                @"\t9.txt", Encoding);
        this.Text = Read.ReadLine();

        quection_count = 0;
        correct_answers = 0;
        wrong_answers = 0;

        array = new String[30];
    }
    catch (Exception)
    {
        MessageBox.Show("No test Found");
    }
    □□□□□□□□();

}
private void button3_Click(object sender, EventArgs e)
{

    radioButton1.Visible = true;
    radioButton2.Visible = true;
```

```
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start();
    return;

}
private void button4_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start2();
    return;

}
private void button5_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start3();
    return;

}
private void button6_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start4();
    return;

}
private void button7_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start5();
    return;

}
private void button8_Click(object sender, EventArgs e)
{
```

```
    start6();
    return;

}
void start6()
{
    var Encoding = System.Text.Encoding.GetEncoding(65001);
    try
    {

        Read = new System.IO.StreamReader(
        System.IO.Directory.GetCurrentDirectory() +
                                @"\t6.txt", Encoding);
        this.Text = Read.ReadLine();

        quection_count = 0;
        correct_answers = 0;
        wrong_answers = 0;

        array = new String[30];
    }
    catch (Exception)
    {
        MessageBox.Show("□□□□□□□ □□□□□ □□□□□");
    }
    □□□□□□□();

}
private void label2_Click(object sender, EventArgs e)
{

}

private void □□□□1ToolStripMenuItem_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start();
    return;

}

private void □□□□2ToolStripMenuItem_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start2();
    return;
```

```
}

private void □□□□3ToolStripMenuItem_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start3();
    return;

}

private void □□□□4ToolStripMenuItem_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start4();
    return;

}

private void □□□□5ToolStripMenuItem_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start5();
    return;

}

private void □□□□6ToolStripMenuItem_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start6();
    return;

}

private void □□□□7ToolStripMenuItem_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
```

```
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start7();
    return;

}

private void □□□□8ToolStripMenuItem_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start8();
    return;


}

private void □□□□9ToolStripMenuItem_Click(object sender, EventArgs e)
{
    radioButton1.Visible = true;
    radioButton2.Visible = true;
    radioButton3.Visible = true;
    label1.Visible = true;
    button1.Visible = true;
    start9();
    return;

}

private void □□□□□□□ToolStripMenuItem_Click(object sender, EventArgs e)
{
    start0();
    return;
    □□□□□□□0();

}
void start0()
{
    var Encoding = System.Text.Encoding.GetEncoding(65001);
    try
    {

        Read = new System.IO.StreamReader(
        System.IO.Directory.GetCurrentDirectory() +
                                @"\t0.txt", Encoding);
        this.Text = Read.ReadLine();
```

```
    }
    catch (Exception)
    {
        MessageBox.Show("□□□□□□□□ □□□□□□□ □□□□□□");
    }
    □□□□□□□□0();

}
void □□□□□□□□0()
{
    label3.Text = Read.ReadLine();
                          =
    label32.Text = Read.ReadLine();


    label3.Visible= true;
                          =
    label32.Visible = true;

    radioButton1.Visible = false;
    radioButton2.Visible = false;
    radioButton3.Visible = false;
    label1.Visible= false;
    button1.Visible = false;




}
void starthelp()
{
    var Encoding = System.Text.Encoding.GetEncoding(65001);
    try
    {

        Read = new System.IO.StreamReader(
        System.IO.Directory.GetCurrentDirectory() +
                                @"\th.txt", Encoding);
        this.Text = Read.ReadLine();




    }
    catch (Exception)
    {
        MessageBox.Show("□□□□□□□□ □□□□□□□ □□□□□□");
    }
    □□□□□□□□0();

}
private void label3_Click(object sender, EventArgs e)
```

```
        {

        }

        private void label4_Click(object sender, EventArgs e)
        {

        }

        private void label5_Click(object sender, EventArgs e)
        {

        }

        private void menuStrip1_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
        {

        }

        private void □□□□□ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void □□□□□□□□□□ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            starthelp();
            return;
            □□□□□□□□0();

        }
    }
}
```