ВІДЗИВ НАУКОВОГО КЕРІВНИКА НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ

Студента Кобрина Андрія Григоровича
представлену на здобуття освітнього ступеня бакалавр
зі спеціальності 123 «Комп'ютерна інженерія»

1. Робота виконана на тему „Навчально-тренувальна програма для набуття навичок з IP-адресації з модулем для контролю знань ".

2. Актуальність роботи зумовлена потребою у розробці та впровадженні у практику діяльності навчальних закладів або для самостійного навчання комп'ютерних програм з отримання ІТ-знань.

3. Основний зміст пояснювальної записки роботи викладено на 55 сторінках тексту, записка містить 72 ілюстрації, 9 сторінок додатків. Список використаних літературних джерел налічує 25 найменувань. Невід'ємною складовою роботи є реалізована за допомогою мови C# у середовищі розробки Microsoft Visual Studio навчально-тренувальна програма.

4. Пояснювальна записка складається з вступу, трьох розділів основного тексту, висновків, списку використаної літератури, додатків.

Розділ 1 присвячений аналізу актуальності розробки програмних продуктів навчального призначення, проведено аналіз існуючих програм та сайтів, їх переваг та недоліків, принципи роботи та можливості розробки за допомогою середовища Microsoft Visual Studio та порівняння її з іншими, розглянута мова C#, як інструмент розробки, а також розглянуті питання створення бази даних за допомогою SQLite, зокрема, розглянуто: поняття навчально-тренувальної програми, структура мови програмування C#. У цьому ж розділі спираючись на здобуту інформації та її аналіз поставлено завдання на розробку.

Розділ 2 присвячений питанням проектування навчально-тренувальної програми та модуля для контролю знань, зокрема, структурі, алгоритму роботи програми. У розділі здійснено вибір інструментів для реалізації програми, розроблені завдання та створена база даних із варіантами для них, розроблено алгоритм оцінювання.

Розділ 3 присвячений питанням дослідження роботи програми та її ефективності. У розіділ також проведено детальний опис інструкції з користування для кожного з варіантів програми.

5. Висновки містять сумарні результати проведеної роботи.

6. Список використаних літературних джерел має достатній для даного виду робіт обсяг, є належно оформленим та опрацьованим у ході виконання роботи.

7. Додатки містять лістинг програмного коду.

8. Таблиці та ілюстрації оформлені досить якісно, є інформативними та наведені у відповідних місцях роботи.

9. Якість оформлення роботи є досить високою.

10. Виявлені на етапі попереднього захисту недоліки оформлення пояснювальної записки та презентації усунені. Рекомендації стосовно покращення програми враховані.

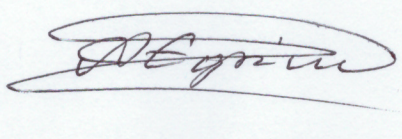
11. Ставлення студента до виконання випускної кваліфікаційної роботи – досить відповідальне.

12. У цілому при написанні випускної кваліфікаційної роботи студент Кобрин А. Г. виявив здібності до оволодіння знаннями, уміннями і навичками з таких дисциплін, як «Комп'ютерні мережі», «Об'єктно-орієнтоване програмування», «Адміністрування та захист баз та сховищ даних». Також студент Кобрин А. Г. показав належні вміння підібрати необхідну літературу та інші інформаційні джерела, проаналізувати їх, виконати постановку задачі на розробку, розробити проект навчально-тренувальної програми та модуля для контролю знань, обрати відповідний інструментарій, створити закінчений цілісний програмний продукт і оформити свої розробки та напрацювання як випускну кваліфікаційну роботу.

Випускна кваліфікаційна робота студента Кобрина А. Г. написана на високому рівні та може бути допущена до захисту в Екзаменаційній комісії.

Пропонована оцінка – «добре».

Науковий керівник
старший викладач кафедри КІ та КБ
15 червня 2020 року



І. А. Оринчак

РЕЦЕНЗІЯ
на кваліфікаційну роботу
студента 4 курсу групи КІ-2 спеціальності 123 «Комп'ютерна інженерія»
факультету інформаційно-комп'ютерних технологій
Державного університету «Житомирська політехніка»
Кобрина Андрія Григоровича
на тему: „Навчально-тренувальна програма для набуття навичок з IP-адресації з модулем для контролю знань"

Кваліфікаційна робота студента Кобрина А. Г. „Навчально-тренувальна програма для набуття навичок з IP-адресації з модулем для контролю знань" присвячена актуальній тематиці – розроблення засобів самостійного навчання та впровадження у освітній процес закладів освіти додаткових інструментів отримання та контролю знань.

Робота містить два програмних модулі (створені за допомогою мови C# та середовища розробки Microsoft Visual Studio, які у процесі роботи використовують базу даних SQLite) та пояснювальну записку наступного складу: вступ, три розділи основного тексту, висновки, список використаної літератури, додатки.

У першому розділі пояснювальної записки приведено аналіз актуальності розробки програмних продуктів навчального призначення, проведено аналіз існуючих програм та сайтів, їх переваг та недоліків, розглянуто принцип роботи та можливості розробки за допомогою середовища Microsoft Visual Studio та середовища БД SQLite. У цьому ж розділі на базі проведено аналізу поставлено завдання на розробку.

У другому розділі пояснювальної записки здійснено проектування основного модуля навчально-тренувальної програми та модуля для контролю знань, зокрема, описано її структуру, алгоритм роботи, здійснено вибір інструментів для реалізації програм, створена база даних із варіантами завдань, розроблено алгоритм оцінювання.

У третьому розділі пояснювальної записки проведено дослідження роботи програми та її ефективності. А також проведено детальна інструкція користувача.

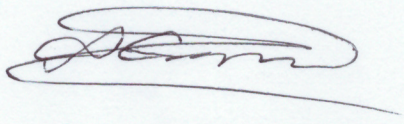
Зміст роботи в цілому відповідає поставленому завданню. Кваліфікаційна робота виконана на належному рівні. Тема роботи розкрита достатньо. Студентом продемонстровані належні фахові компетентності, отримані під час навчання та у процесі виконання роботи. Матеріал пояснювальної записки викладено досить логічно і технічно грамотно. Ілюстративний матеріал (рисунки, таблиці), що наведений у роботі, оформлений якісно і наведений у відповідних місцях. Оформлення роботи у цілому відповідає вимогам стандартів. Програмні модулі реалізовано у середовищі розробки Microsoft Visual Studio на мові програмування C#. Їх функціонал відповідає заявленому.

Позитивною стороною є те, що отримані результати мають практичну цінність – розроблений проект планується до впровадження в діяльності навчальних закладів. Недоліками роботи є наступні: недостатнє пропрацювання питань застосування навчальних та контролюючих програм, відносно програм реалізації програмнох модулів роботи, наявність деяких відхилень від вимог щодо оформлення пояснювальної записки.

У цілому кваліфікаційна робота студента Кобрина А. Г. „Навчально-тренувальна програма для набуття навичок з IP-адресації з модулем для контролю знань " відповідає вимогам рівня вищої освіти «бакалавр» спеціальності 123 «Комп'ютерна інженерія» і рекомендується до захисту в ЕК.

Пропонована оцінка роботи – «добре».

Рецензент,
завідувач кафедри КІ та КБ,
к.т.н
17 червня 2020 р.



А. А. Єфіменко,

Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп’ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему:

«Навчально-тренувальна програми для набуття навичок з IP-адресації з модулем для контролю знань»

Виконав студент 4-го курсу, групи КІ-2
спеціальності 123 «Комп’ютерна інженерія»
_____ Кобрин А. Г.

Керівник, старший викладач кафедри КІ та КБ
_____ Оринчак І. А.

Рецензент, завідувач кафедри КІ та КБ, к.т.н.
_____ Єфіменко А. А.

Житомир 2020

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА КІБЕРБЕЗПЕКИ
СПЕЦІАЛЬНІСТЬ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерної інженерії
та кібербезпеки
Єфіменко А. А.
«06» квітня 2020 р.

ЗАВДАННЯ
на випускну кваліфікаційну роботу

Студента **Кобрина Андрія Григоровича**
Тема роботи: **Навчально-тренувальна програми для набуття навичок з ір-адресації з модулем для контролю знань** затверджена Наказом по університету від **«06» квітня 2020 р. № 98с**
Термін здачі студентом закінченої роботи – **1 червня 2020 р.**
Вихідні дані для роботи:

1. Мова програмування C#
2. База даних SQLite
3. Середовище розробки Microsoft Visual Studio
4. Пакет System.Data.SQLite для Microsoft Visual Studio
5. Операційні системи Windows XP, Windows 7, Windows 10

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Оринчак І. А.	07.04.2020	07.04.20
2	Оринчак І А.	19.04.20	19.04.20
3	Оринчак І А.	12.05.20	12.05.20

Керівник Оринчак І. А.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка завдання	7 квітня 2020 – 8 квітня 2020	Виконано
2	Пошук, огляд та аналіз літературних джерел щодо організації комп'ютерної програми навчального призначення	9 квітня 2020 – 11 квітня 2020	Виконано
3	Аналіз технологій комп'ютерних програми навчального призначення	12 квітня 2020 – 16 квітня 2020	Виконано
4	Аналіз можливостей мови програмування C# для організації навчально-тренувальної програми	17 квітня 2020 – 19 квітня 2020	Виконано
5	Розробка загального проекту навчально-тренувальної програми	20 квітня 2020 – 24 квітня 2020	Виконано
6	Розробка алгоритмів генерації даних для завдань	25 квітня 2020 – 30 квітня 2020	Виконано
7	Створення бази даних із варіантами для завдань	1 травня 2020 – 4 травня 2020	Виконано
8	Розробка алгоритмів оцінювання кожного завдання	5 травня 2020 – 15 травня 2020	Виконано
9	Розробка додатково функціоналу в заданнях та програмі в цілому	16 травня 2020 – 19 травня 2020	Виконано
11	Тестування роботи навчально-тренувальної програми	20 травня 2020 – 23 травня 2020	Виконано
12	Аналіз результатів тестування	24 травня 2020 – 25 травня 2020	Виконано
13	Написання пояснювальної записки	26 травня 2020 – 29 травня 2020	Виконано
14	Підготовка презентації та виступу	30 травня 2020 – 31 травня 2020	Виконано
15	Отримання відгуку керівника та рецензії на кваліфікаційну роботу	9 червня 2020 – 13 червня 2020	Виконано
16	Захист	20 червня 2020	Виконано

Студент _____ Кобрин А. Г.

Керівник _____ Оринчак І. А.

РЕФЕРАТ

Кваліфікаційна робота складається з проекту навчально-тренувальної програми для набуття навичок з IP-адресації з модулем для контролю знань складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаних джерел, додатків і має 55 сторінки основного тексту, 72 ілюстрації та 9 сторінок додатків. Список використаних джерел містить 25 найменувань і займає 2 сторінки. Загалний обсяг робти - 72 сторінок.

Метою роботи є розробка навчально-тренувальної програми «IPTeaster» та модулю для контролю знань «IPTeacher».

В роботі визначено основні завдання на розробку програм, які дозволять користувачам вивчати IP-адресацію і набувати навчики у обрахунку параметрів IP-адрес. Також в роботі проаналізовані інші подібні програми та сайти навчального призначення, розглянута мова C# і Microsoft Visual Studio, як інструмент та середовище для розробки. Розроблені навчально-тренувальна програма та модуль для котролю знань. Розроблена єдина база даних SQLite з варіантами завдань для кожної з програм Проведений детальний інструктаж користування кожною програмою. Також наводяться результати тестування роботи розроблених програм, перевірка алгоритмів, функціоналу кожного елемента тощо.

КЛЮЧОВІ СЛОВА: НАВЧАЛЬНО-ТРЕНУВАЛЬНА ПРОГРАМА, МОВА C#, MICROSOFT VISUAL STUDIO, БАЗА ДАНИХ SQLITE, «IPTESTER», «IPTEACHER»

РЕФЕРАТ

Кваліфікаційна робота складається з проекту навчально-тренувальної програми для набуття навичок з IP-адресації з модулем для контролю знань складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаних джерел, додатків і має 55 сторінки основного тексту, 72 ілюстрації та 9 сторінок додатків. Список використаних джерел істить 25 найменувань і займає 2 сторінки. Загалний обсяг робти - 72 сторінок.

Метою роботи є розробка навчально-тренувальної програми «IPTeaster» та модулю для контролю знань «IPTeacher».

В роботі визначено основні завдання на розробку програм, які дозволять користувачам вивчати IP-адресацію і набувати навчики у обрахунку параметрів IP-адрес. Також в роботі проаналізовані інші подібні програми та сайти навчального призначення, розглянута мова C# і Microsoft Visual Studio, як інструмент та середовище для розробки. Розроблені навчально-тренувальна програма та модуль для котролю знань. Розроблена єдина база даних SQLite з варіантами завдань для кожної з програм Проведений детальний інструктаж користування кожною програмою. Також наводяться результати тестування роботи розроблених програм, перевірка алгоритмів, функціоналу кожного елемента тощо.

КЛЮЧОВІ СЛОВА: НАВЧАЛЬНО-ТРЕНУВАЛЬНА ПРОГРАМА, МОВА C#, MICROSOFT VISUAL STUDIO, БАЗА ДАНИХ SQLITE, «IPTESTER», «IPTEACHER»

ABSTRACT

The qualification work consists of a draft training program for the acquisition of skills in IP-addressing with a module for knowledge control consists of a list of abbreviations, introduction, three sections, conclusions, list of sources, appendices and has 55 pages of main text, 72 illustrations and 9 application pages. The list of used sources has 25 names and occupies 2 pages. Total workload – 72 pages.

The purpose of the work is develop a training program «IPTeaster» and a module for knowledge control «IPTeacher».

The paper identifies the main tasks for the development of programs that will allow users to learn IP-addressing and acquire skills in calculating the parameters of IP-addresses. Other similar programs and educational sites are also analyzed, the C # language and Microsoft Visual Studio are considered as a tool and environment for development. A training program and a module for knowledge control have been developed. A single SQLite database with task variants for each of the programs has been developed. Detailed instructions for using each program have been provided. Also the results of testing of work of the developed programs, check of algorithms, functionality of each element, etc. are resulted.

KEY WORDS: TRAINING PROGRAM, LANGUAGE C#, MICROSOFT VISUAL STUDIO, SQLITE DATABASE, IPTESTER, IPTEACHER

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 7
ВСТУП 8
РОЗДІЛ 1. ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ 10
1.1 Обґрунтування актуальності теми роботи по розробці навчально-тренувальної програми для набуття навичок з ір-адресації. 10
1.2. Аналіз особливостей навчально-тренувальної програми. 13
1.3. Аналіз вимог до навчально-тренувальної програми. 16
1.4. Аналіз можливостей мови програмування C# щодо розробки навчально-тренувальної програми. 17
1.5. Постановка завдання на розробку. 21
Висновки з першого розділу. 24
РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ, ВИБІР ІНСТРУМЕНТІВ ДЛЯ C# ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ. 24
2.1. Розробка структури програми 24
2.2. Вибір інструментів C# для реалізації навчально-тренувальної програми. 35
2.3. Розробка програмного забезпечення. 36
Висновки з другого розділу 40
РОЗДІЛ 3. ДОСЛІДЖЕННЯ ПРАЦЕЗДАТНОСТІ ТА ІНСТРУКЦІЯ КОРИСТУВАННЯ ПРОГРАМОЮ 40
3.1. Дослідження працездатності та ефективності 40
3.2 Інструкція користувачу. 50
Висновки по 3 розділу 58
ВИСНОВКИ 59
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 60
ДОДАТКИ 62

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – База даних

MVS – Microsoft Visual Studio

OS – Operation System

ПЗ – Програмне забезпечення

SQL – Structured Query Language

WEB – Інтернет-простір

ПК – Персональний комп’ютер

ЕЗНП – Електронні засоби навчального призначення

CLR – Common Language Runtime

ВСТУП

Актуальність теми. У зв'язку з тим, що сьогодні з'являються і стрімко розвиваються новітні технології, інтернет, поступово традиційне навчання втрачає свою популярність. Носити з собою велику кількість підручників чи довідників не зовсім практично, адже, для того, щоб отримати ту чи іншу інформацію достатньо скористатись інтернет-мережею або ж завантажити книгу чи будь-яке джерело інформації в електронному вигляді на ПК або інші гаджети. Зараз особливо актуальними є комп'ютерні програми навчального призначення та освітні сайти. Це означає що завдяки навчально-тренувальним програмам економиться час викладача, а інформація і завдання у вільному доступі. Також, відкриваються онлайн школи, де навчання відбуваються в онлайн режим, зокрема, з використанням програм та сайтів начального призначення.

Таким чином, процес навчання повинен іти в ногу з часом, а навчально-тренувальні програми в повній мірі задовольняють цю потребу.

Вміння спроектувати та реалізувати авторську навчально-тренувальну програму значною мірою може вирішити проблему варіативності базових, профільних або елективних занять для полегшення роботи та широкого спектру інтересів студентів.

Метою кваліфікаційної роботи є розробка алгоритмів роботи та реалізація авторських навчально-тренувальної програми «IPTeaster» та модулю для контролю знань «IPTeacher».

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– обґрунтувати актуальність теми роботи по розробці навчально-тренувальної програми для набуття навичок з ір-адресації;

– проаналізувати особливості навчально-тренувальної програми та модулю для контролю знань;

– проаналізувати вимоги до навчально-тренувальної програми та модулю для контролю знань;

– проаналізувати можливості мови програмування C# щодо розробки навчально-тренувальної програми;

– розробити структуру програми «IPTeaster» та модулю «IPTeacher»;

– вибрати інструменти C# для реалізації навчально-тренувальної програми;

– розробити програмне забезпечення;

– дослідити працездатність та ефективність модулю контролю знань та навчально-тренувальної програми;

– створити інструкцію по користуванню навчально-тренувальною програмою та модулем контролю знань.

Об'єктом дослідження є авторська навчально-тренувальну програму «IPTeaster» та модуль для контролю знань «IPTeacher».

Предметом дослідження стали особливості створення навчально-тренувальної програми «IPTeaster» та модулю для контролю знань «IPTeacher».

Аналіз практичних значень отриманих результатів показує, що навчально-тренувальна програма та модуль для контролю знань в повній мірі можливо використовувати для здобуття навичок з ір-адресації, а отже, цей комплекс програм може використовуватись не тільки в навчальних закладках (школи, ліцеї, університети тощо), а й для власного користування окремими користувачами. Також, цей комплекс цілком може бути інтегрованим на будь-який освітній сайт.

В основу методичної бази лягли такі загальнонаукові методи, як теоретичний аналіз, методи системного аналізу, методи об’єктно-орієнтованого проектування, веб-програмування та дизайну. З емпіричних методів були використані порівняльно-типологічний аналіз і дискурс-аналіз.

РОЗДІЛ 1. ОБҐРУНТУВАННЯ АКТУАЛЬНОСТІ ТЕМИ РОБОТИ

1.1 Обґрунтування актуальності теми роботи по розробці навчально-тренувальної програми для набуття навичок з ір-адресації.

Навчальні програми використовуються практично у всіх сферах, зокрема і у навчальних закладах. В широкому розумінні терміну «навчальна програма» – це документ, що окреслює коло знань, умінь та навичок за виділений час (години на тему, навчальний рік тощо). Ціль таких програм – надання знань тим чи іншим методом. Самі методи передбачені предметом, темою чи конкретним завданням або ціллю. Навчально-тренувальні програми можуть бути у вигляді підручника, сайту, комп’ютерної програми, довідника, ЕЗНП тощо. ЕЗНП – навчальні засоби, які зберігаються на носіях даних і відтворюються на електронному обладнанні (комп'ютерні програми, електронні таблиці, електронні бібліотеки, тестові завдання, тощо) .

Проте, носити з собою велику кількість підручників чи довідників не зовсім практично, адже, для того, щоб отримати ту чи іншу інформацію достатньо скористатись інтернет-мережею або ж завантажити книгу чи будь-яке джерело інформації в електронному вигляді на ПК або інші гаджети. Так само на свій смартфон чи ПК можна завантажити і навчальні програми та тренажери, які завжди будуть під рукою.

В епоху комп’ютерних технологій та мереж особливо актуальними є комп’ютерні програми навчального призначення та освітні сайти. З доступом до інтернет-мережі будь-яка інформація також стає доступною. Проте, варто розуміти – її настільки багато, що загубитись в такому потоці дуже легко. До того ж, цей потік постійно зростає. В такому випадку на допомогу приходять комп’ютерні програми та сайти.

Знання про способи та засоби організації навчально процесу в умовах користування ПК називають інформаційною технологією навчання. Реалізація цієї технології припускає наявність ПК, як технічного засобу навчання, і відповідного прикладного ПЗ для організації навчального процесу.

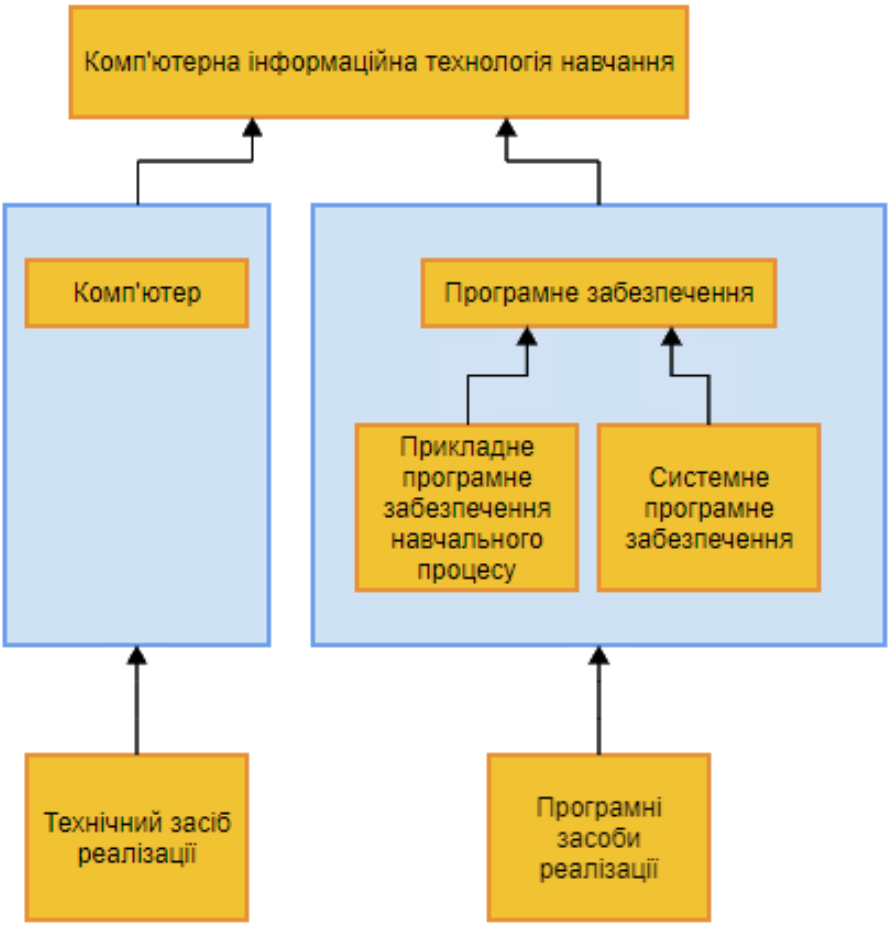


Рис.1.1. Структура інформаційної техногології навчання

Комп’ютерні програми навчального призначення можна класифікувати по типам наступним чином:

– Комп’ютерні підручники;

– Навчальні програми;

– Навчально-ігрові програми;

– Тренажери;

– Предметно-орієнтовані середовища (моделюючі програми, навчальні пакети тощо);

– Контролюючі програми;

– Лабораторні практикуми;

– Довідники, бази даних навчального призначення.

Проте, в програмах здебільшого наявні властивості, особливості та функції декількох типів одночасно. Наприклад, в програмі надається інформація щодо вирішення задачі, далі слідує тренажер із цією задачею, а наприкінці модуль контролю, який перевіряє правильність відповідей. І це все в одній програмі. Також, часто зустрічаються програми розбиті по типам, але пов'язані між собою одною темою, предметом чи ціллю. Вище згаданий приклад програми можна розбити на 2, при цьому цілісність навчання залишиться такою ж. Тобто, в одну програму можна вмістити теоретичний матеріал, а в іншій по викладеній темі завдання з модулем, який буде перевіряти правильність відповідей. Сайти працюють за таким же ж способом або ж подібним. Тільки всі ці дії відбуваються у WEB-браузері. Зазвичай сайти містять всю потрібну інформацію на тому ж вікні що і завдання, тобто організовані як одна програма. Проте, зустрічаються і тренувальні сайти, а потрібна інформація викладається як посилання на інші сайти.

Досить часто можна зустріти симбіоз сайтів та комп'ютерних програм навчального призначення. Викладення інформації відбувається на самому сайті, а тренажер створений у вигляді програми.

Оскільки, навчання таким способом передбачає користування сайтом чи програмою, а навіть і двома варіантами, то окрім знань що передбачені темою, користувач буде отримувати навички сайтової та програмної діяльності, а також підвищить сайтову та програмну компетентність, яка неабияк важлива у сучасному світі.

В кожній навчальній програмі перед користувачем ставиться конкретна ціль і надається все необхідне для її досягнення. Весь матеріал складений таким чином, що користувач вивчає тему за темою.

Актуальність сайтів та програм навчального призначення також зумовлена їхньою доступністю та зручністю. Все більше книг скануються і викладаються в електронному вигляді, для того, щоб кожен бажаючий міг отримав до них доступ. Сайти та програми навчального призначення можна назвати альтернативою традиційному навчанню в школах, університетах тощо,

оскільки вони є самостійними. Самостійність полягає в тому, що достатньо провести інструкцію користування програмою, а теоретичний матеріал, завдання і їх оцінку програма виконає самостійно. Це означає, що завдяки навчально-тренувальним програмам економиться час викладача, а інформація і завдання завжди у вільному доступі. Також, відкриваються онлайн школи, де навчання відбуваються в онлайн режим, зокрема, з використанням програм та сайтів начального призначення. Нещодавній карантин довів, що такі програми неабияк затребувані.

Таким чином, процес навчання повинен іти в ногу з часом, а навчально-тренувальні програми в повній мірі задовольняють цю потребу.

1.2. Аналіз особливостей навчально-тренувальної програми.

Для аналізу особливостей навчально-тренувальних програм були обрані сайт NetAcad (Network Academy) компанії Cisco і навчально-ігровий комплекс у вигляді комп'ютерної програми Астрофізик++. На їх прикладі зручно розглянути всі особливості комп'ютерних програм та сайтів навчального призначення, адже решта влаштовані подібним чином.

NetAcad – продукт компанії Cisco.

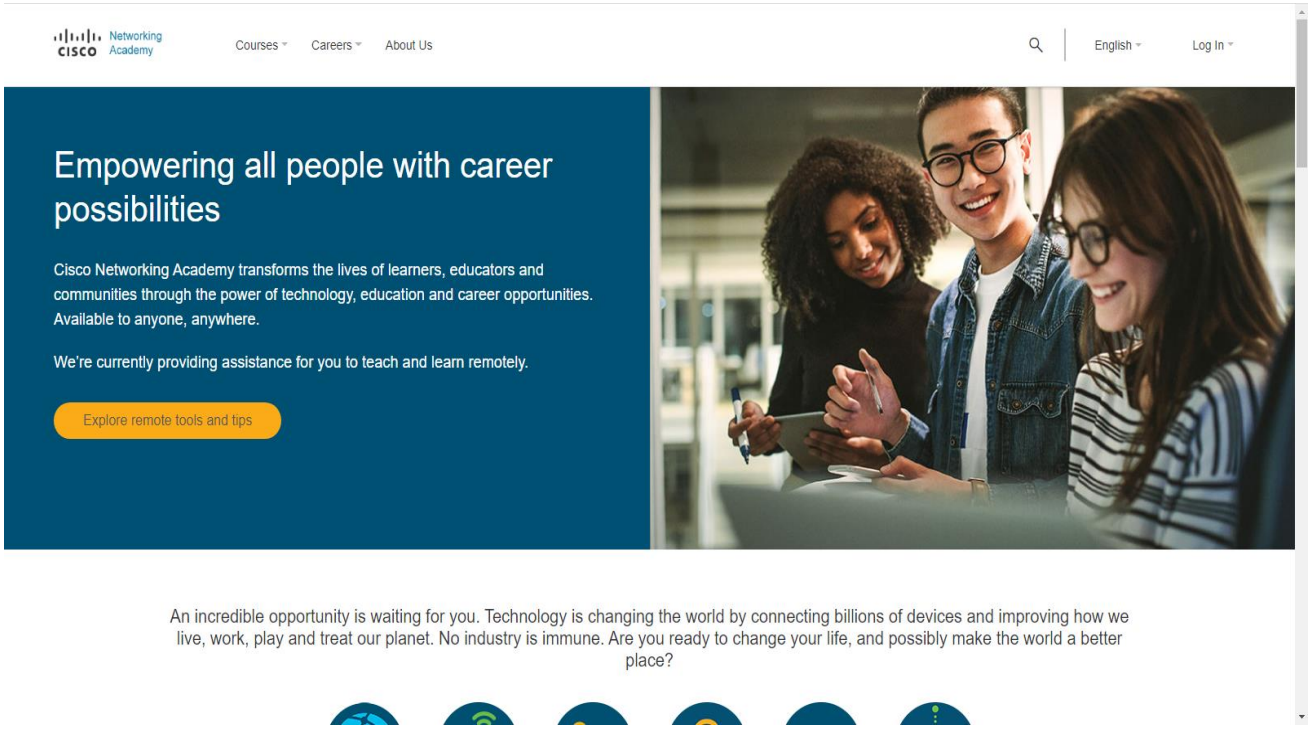


Рис.1.2. Вигляд головної сторінки NetAcad

Це сайт навчального призначення в якому зібрані теоретичні та практичні матеріали, що напряму пов’язані із мережами та розбиті по курсам. Це яскравий приклад симбіозу сайту та програми навчального призначення. Кожен курс містить розділи із теоретичним матеріалом та практичними завданнями. Також передбачені практичні і теоретичні тести по всьому матеріалу курсу. Практичні завдання відбуваються у програмі Cisco Packet Tracer, яку можна завантажити із сайту NetAcad. Після завершення сайт оцінює успішність здачі екзамену, тесту чи практичного завдання. Для проходження тестової частини і завдань виділяється певний час, за який користувачеві потрібно виконати завдання чи тест. Також, у користувача є можливість відслідковування прогресу навчання. NetAcad надає можливість отримати сертифікат з певного курсу, що є вагомою перевагою сайту. Для цього потрібно завершити успішно курс і пройти сертифікаційний екзамен.

Сертифікації Cisco – це професійні сертифікації в області інформаційних технологій для роботи з продуктами Cisco Systems. Існує 5 рівнів сертифікації:

- Entry;
- Associate;
- Professional;
- Expert;
- Architect.

А також є окрема сертифікація спеціаліст. Найвищий ступінь – Architect (CCAr).

Щоб почати проходження певного курсу потрібно зареєструватись на сайті і на кожен курс окремо. Користувач може бути зареєстрованим на декілька курсів одночасно.

Основною перевагою сайтів є адаптивність. Тобто, користувач може мати доступ до сайту і з комп’ютера, і зі смартфона, планшета і тд. Потрібно лиш

мати підключення до інтернет-мережі. А ось неможливість роботи в режимі офлайн є основним недоліком сайтів.

Навчально-ігровий комплекс Астрофізик++.

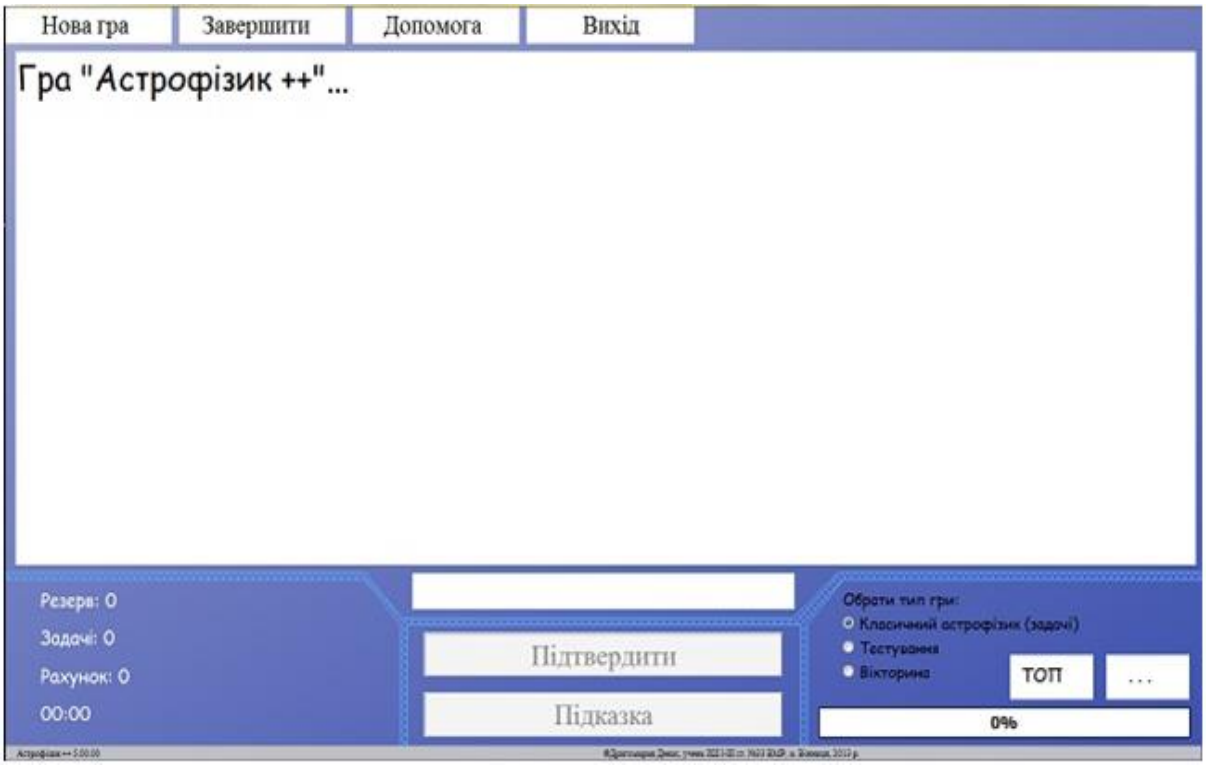


Рис.1.3. Вигляд головного вікна Астрофізик++

Це комплекс для вдосконалення навичок вирішення астрономічних задач та перевірки знань користувачів у тестовому режимі. Навчання відбувається у ігровій формі. Першим кроком є обрання режиму гри у правому нижньому кутку екрана: класичного астрофізика (задачі), тестування або вікторини. Комплекс використовує базу даних із завданнями, що дозволяє зручно додавати нові завдання або ж видаляти чи змінювати застарілі. Також, як і в NetAcad, присутнє оцінювання успішності проходження завдань.

Переваги і недоліки програм протилежні сайтам. Здатність працювати офлайн, безперечно, є основною перевагою комп'ютерних програм. Хоч інтернет-мережа досить поширена, не кожен має доступ до неї. В свою чергу недолік – одноплатформенність. Це означає що якщо програма написана для комп'ютерів на операційній системі Windows, то і запускатись вона буде тільки на ній. Для того щоб користувач мав доступ до такого типу програм на інших

пристроях і операційних системах, потрібно створювати цю ж програму для кожної платформи окремо.

Отже, незалежно від того у вигляді сайту чи програми є комплекс, їх об’єднують спільні риси та особливості, які повинні бути реалізовані у кожній навчально-тренувальній програмі.

1.3. Аналіз вимог до навчально-тренувальної програми.

Основними вимогами до навчально-тренувальних програм, насамперед, є їх затребуваність та актуальність. Мета таких програм та комплексів полягає в тому, щоб донести знання користувачам, учням, студентам тощо. Надати їм навички та уміння. Отже, матеріал, перш за все, повинен бути доцільним та достовірним, зрозумілим та структурованим. Подання інформації повинно відбуватися або до завдань, або паралельно, тобто, в сусідньому вікні. В другому випадку користувач зможе звернутись до теорії одразу, що позволить звіряти правильність дій, якщо це тренувальні завдання, а не тестування. Посилання на теоретичний матеріал у тестуванні неможливо, адже тоді нема сенсу його проводити. Якщо матеріал у великих об’ємах, його потрібно розділити на теми та підтеми, якщо це можливо. Це дозволить користувачу проходити матеріал поступово, від теми до теми, і, відповідно, швидше його вивчити. Задачі, вправи, завдання тощо, які реалізовані в програмі повинні напряму перетинатись із викладеною інформацією, тобто, повинна бути не тільки теоретичний матеріал, що стосується предмету, а й методичні рекомендації щодо виконання завдань. Якщо матеріал поділений на теми, то і завдання теж повинні бути розділені і стосуватись конкретної теми. Інтерфейс повинен бути зрозумілим, перевагою буде зробити його приємним на вигляд або ж нейтральним, тому що різкі, яскраві кольори можуть відволікати користувача або ж йому буде важче сконцентруватися.

Кожен елемент, якщо це потрібно, повинен мати підпис, який буде орієнтувати користувача. Якщо це поле відповіді, то потрібно вказати, якого типу значення в нього потрібно вводити. Якщо ж полів для відповіді декілька, то їм також потрібно надати назву, щоб користувач знав, який параметр туди

вводити. Якщо кнопка, то назвати її, щоб користувач розумів, що вона виконує. Наприклад, кнопка «Оцінка», оцінює завдання.

При контролі знань чи тестуванні потрібно додати таймер, який буде відраховувати час виданий користувачу на розв'язок завдання.

Перевірка завдань повинна відбуватись коректно. Оцінювання завдань в кожній програмі може відбуватися по-різному, тому потрібно вибрати спосіб, який найбільше доречний в залежності від типу завдань. Також, обо'язково повинен бути організований інструктаж користування програмою (проведений напряму або у вигляді тексту тощо).

1.4. Аналіз можливостей мови програмування C# щодо розробки навчально-тренувальної програми.

Мова програмування C# доволі універсальна. Її функціонал разом із бібліотеками і фреймворками (їх число досягає декількох сотень) дозволяють реалізувати практично будь-які ідеї. Вона була розроблена в 1998 – 2001 роках інженерами компанії Microsoft, для платформи Microsoft .NET Framework. Згодом була стандартизована як ECMA–334 і ISO/IEC 23270.

На C# розробляють:

- Додатки для WEB;
- Ігрові програми;
- Додатки для платформ Андроїд або iOS;
- Програми для Windows.

Прибирання сміття – можливість C#, що дозволяє в автоматичному режимі очистити пам'ять від об'єктів, які не використовуються або знищених додатків. Також, в C# наявний інструмент «Обробка винятків». За допомогою нього можна легко виявляти і оброблювати помилки в коді. Сам спосіб є структурованим із широким набором функцій. Компанія Microsoft приділяє значну увагу щодо підтримки цієї мови програмування. А тому регулярно з'являються оновлення і доповнення, виправляються виявлені неполадки

компілятора, розширюються бібліотеки. Таким чином C# постійно розвивається і надалі буде на високих позиціях рейтингу популярних мов програмування.

В рамках платформи до обробки коду підключається CLR – загальномовне виконуюче середовище бібліотек і класів. Це середовище було розроблене Microsoft і є реалізацією світового стандарту Common Language Infrastructure (CLI).

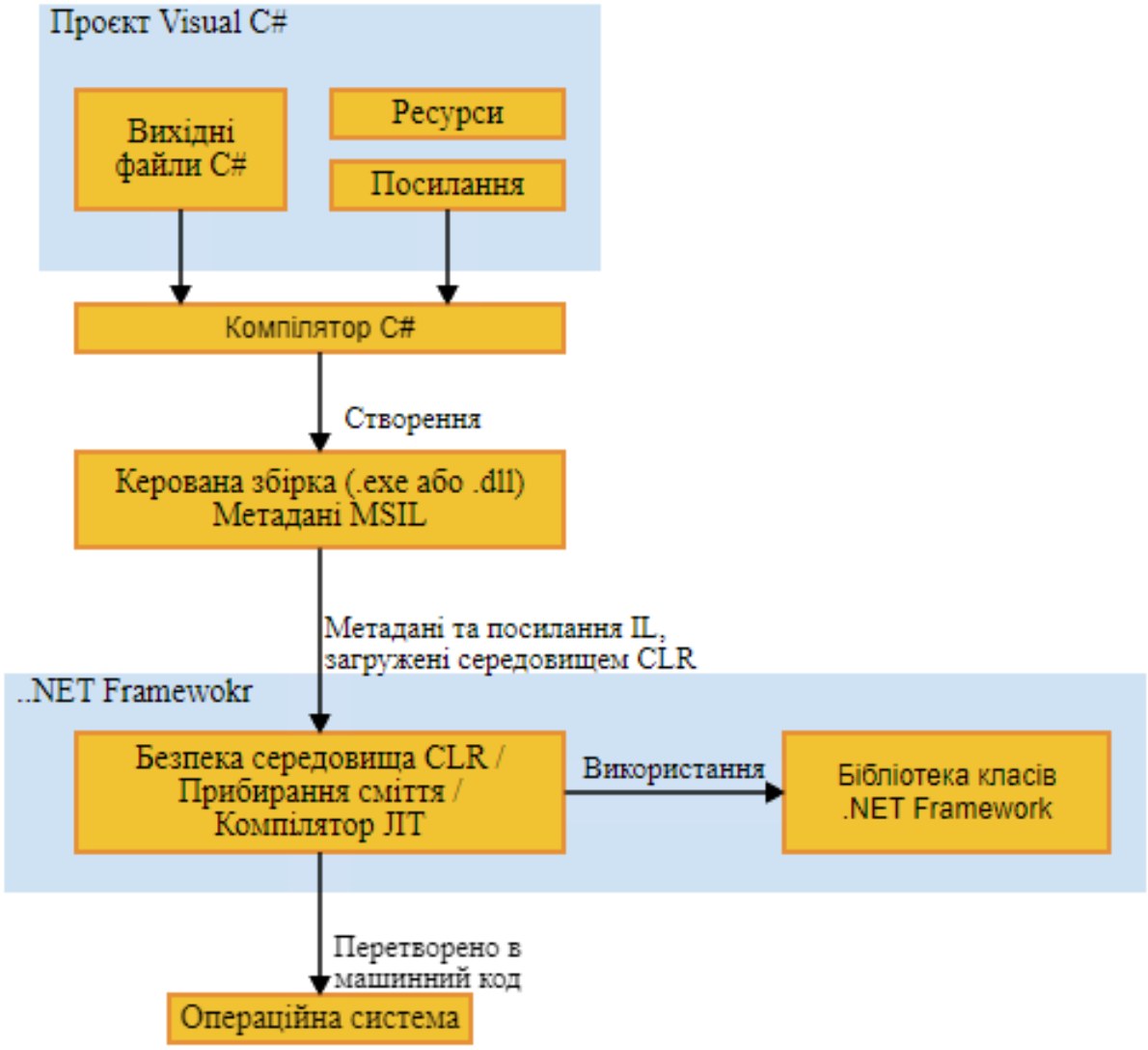


Рис.1.4. Архітектура платформи

Синтаксис C# практично не відрізняється від мов програмування сімейства C. C# протягом довгого часу впевнено лідирує в рейтингу кращих і найбільш затребуваних на ринку розробки мов. Спочатку їм зацікавилися тільки розробники, які пишуть програми під Windows. Але в процесі розвитку

C# інтегрували на Mac, Linux, IOS і Android. Після того, як код платформи відкрили для всіх бажаючих, були зняті практично всі можливі обмеження в застосуванні C#. В результаті мова активно розвивається та застосовується все ширше.

Перед тим, як почати працювати із C#, потрібно обрати середовище розробки. Їх доволі багато, проте найпопулярнішими вважаються Visual Studio, SharpDevelop та MonoDevelop.

SharpDevelop – це безкоштовне середовище розробки для C#, .NET, IronPython, тощо, яке підтримується тільки на платформі Windows. Зазвичай SharpDevelop використовується як альтернатива MVS.

MonoDevelop також безкоштовне середовище розробки, передусім націлене на розробку програм, які використовують і Mono, і Microsoft. NET Framework. На даний момент підтримуються мови C#, Java, Boo, C, C++, Python, тощо. Підтримується MonoDevelop на платформах Linux, Windows та macOS X. Visual Studio підтримується на платформах Windows та macOS.

Нижче наведені вигляди згаданих середовищ:

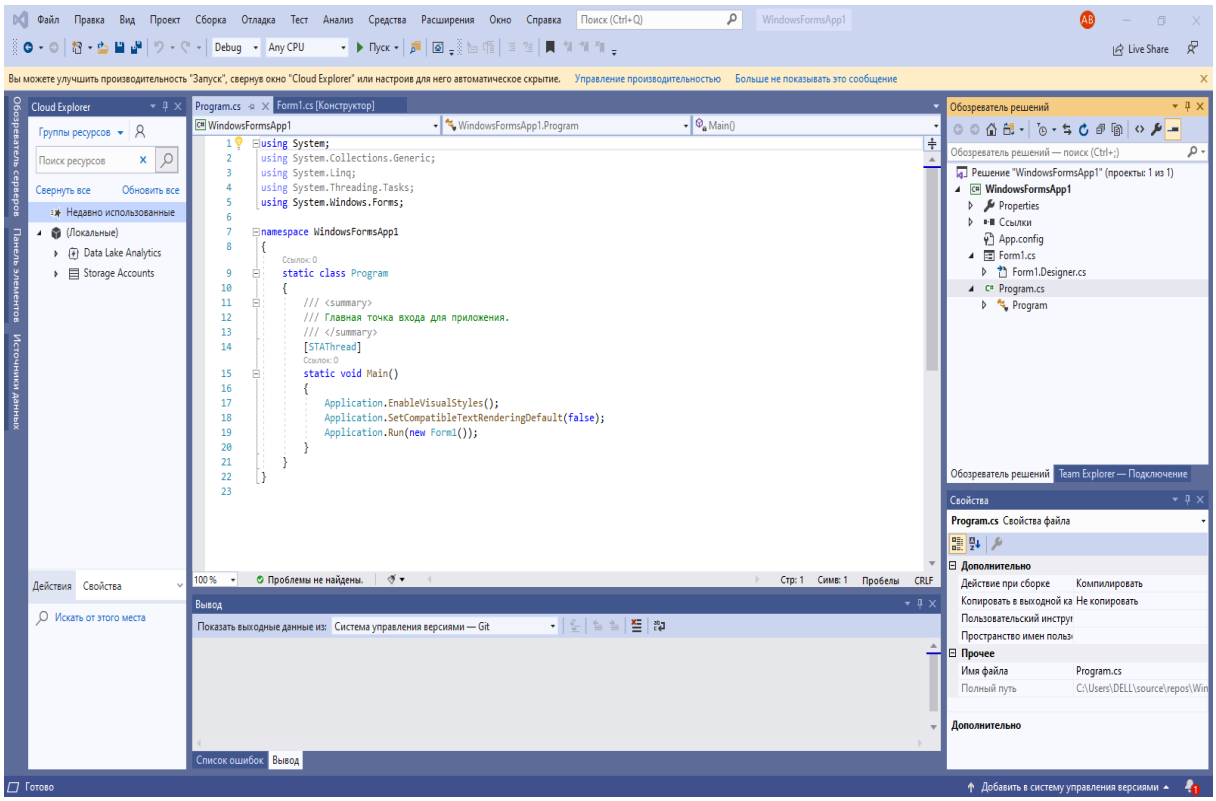


Рис.1.5. Вигляд робочого поля Visual Studio

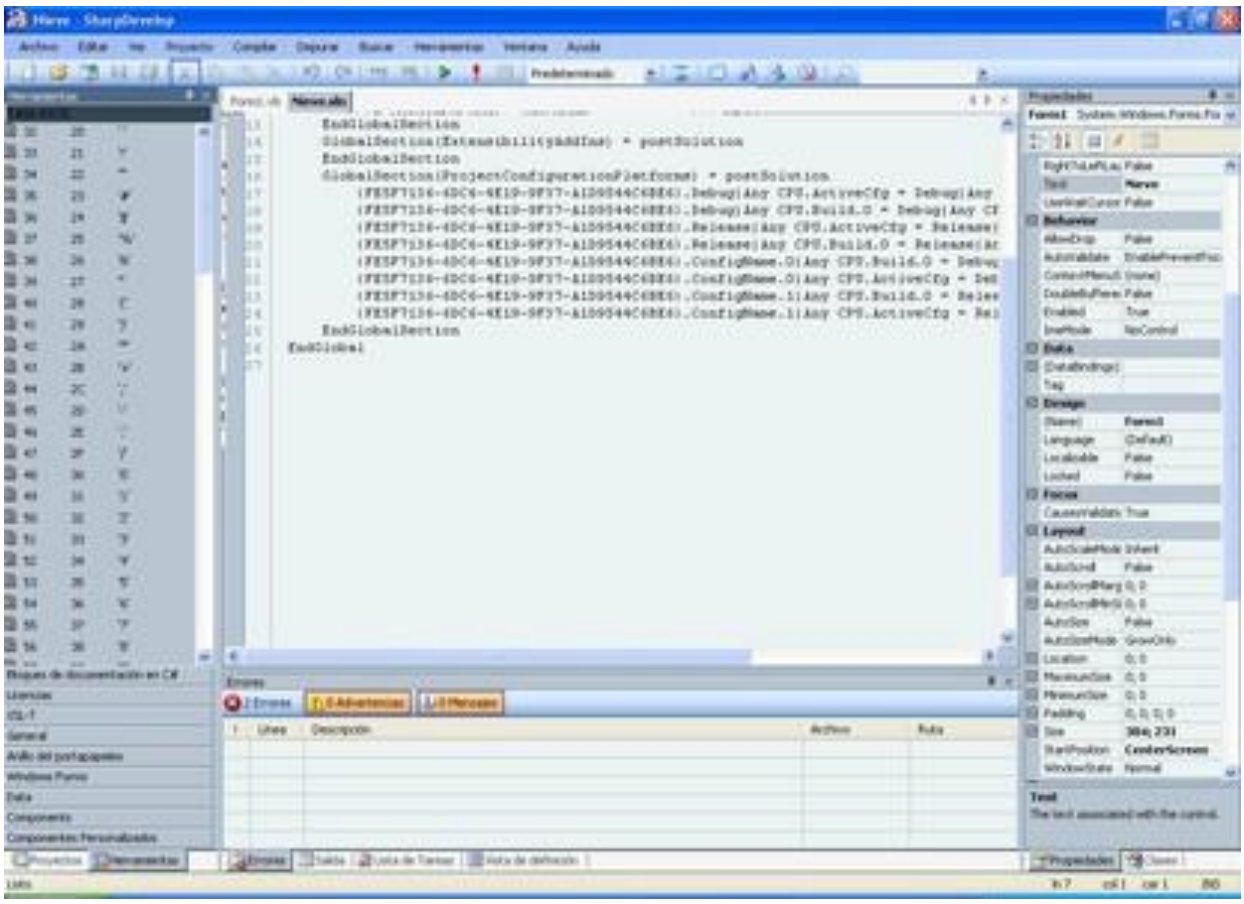


Рис.1.6. Вигляд робочого поля SharpDevelop

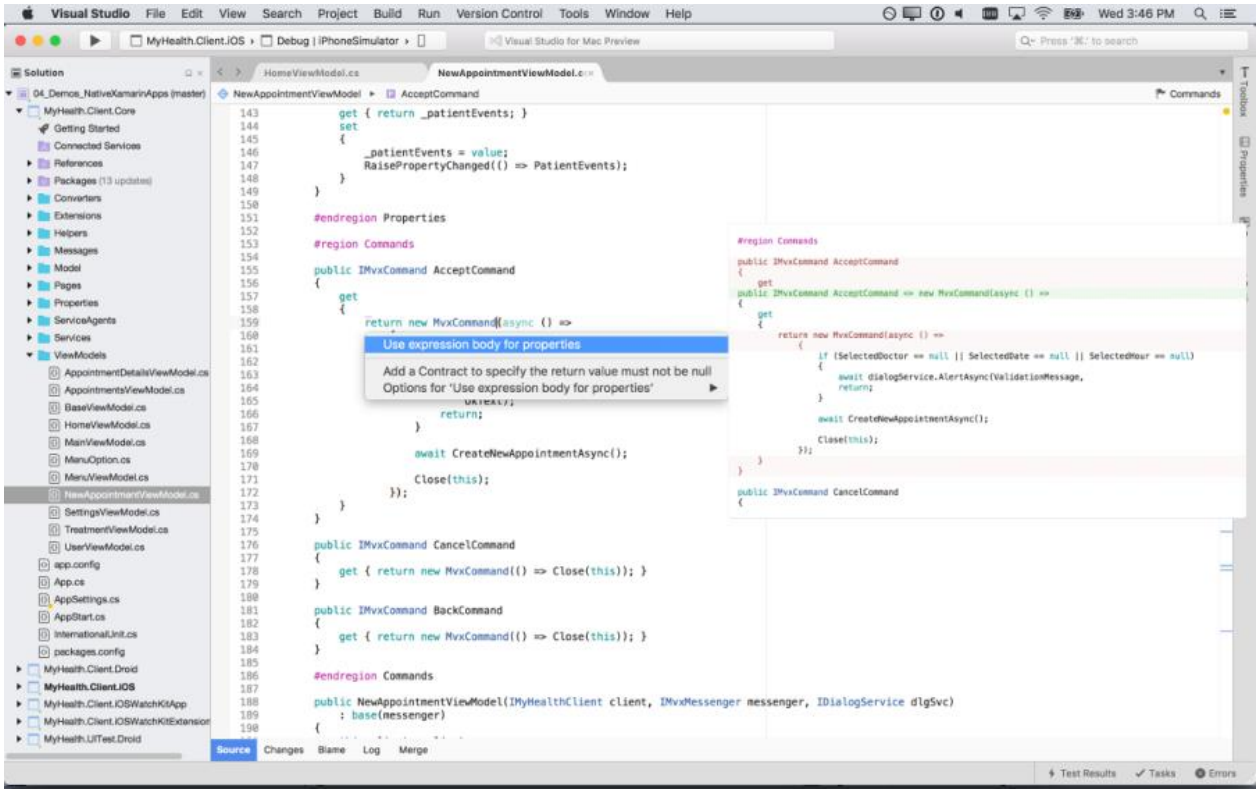


Рис.1.7. Вигляд робочого поля MonoDevelop

Як показано на Рис. 1.5-7. – всі 3 програми подібні між собою. Проте в SharpDevelop, як і в MonoDevelop наявний суттєвий недолік. В цих середовищах відсутній візуальний конструктор форм. В свою чергу, у MVS цей конструктор наявний.

Конструктор дозволяє створювати форми просто перетягуючи потрібні компоненти на робоче поле, що значно полегшує роботу. Це дозволяє з самого початку створення програми бачити її вигляд і зручно його редагувати. Також, є можливість редагування властивостей кожного компоненту, такі як: колір, шрифт, розмір (не в кожному елементі) тощо. При подвійному натисканні комп'ютерною мишею на елемент робочого поля, код автоматично додається в метод «обробник події», і курсор установлюється в його тілі. Тобто, всі повторювані операції виконуються автоматично, залишається лише написати код. Завдяки такій можливості конструктора форм вдається відділити створення макет-вікна від логічної частини програми. Для desktop-програми такий спосіб розробки є найефективнішим і найшвидшим. Також у Visual Studio можливо підключати бібліотеки. Завдяки ним значно збільшуються можливості розробки.

Таким чином, проаналізувавши доступні середовища розробки, була вибрана Visual Studio, оскільки в ній найзручніше створювати desktop-програми.

1.5. Постановка завдання на розробку.

1.5.1. Постановка на завдання навчально-тренувальної програми.

«IPTeacher» – це навчально-тренувальний комплекс, ціллю якого є навчити користувача базовим знанням з ip-адресації.

Оскільки, основна мета – це надання користувачу знань по ip-адресації, то в навчально-тренувальній частині насамперед повинна бути достовірна інформація щодо завдань і теми ip-адресації в цілому або ж посилання на джерела. Також в ній же ж після ознайомлення у користувача повинна бути можливість відпрацювати вивчений матеріал та практичні навички. Оскільки,

інформація про ір-адресацію досить обширна, важливо розділити її по темам, що полегшить навчання.

Також, важливо організувати і практичну частину, в якій користувач зможе відпрацювати вивчений матеріал, який напряму повинен відповідати викладеній теорії. Практична частина повинна бути чітко сформульована, щоб користувач розумів, що від нього потрібно. Сама ж реалізація завдань повинна бути правильно організована. Мається на увазі, що за потреби можливо доведеться розробити базу даних варіантів завдань окремо. Це дозволить легко її змінювати, додавати або ж видаляти варіанти.

Щоб отримати дані для розв'язку завдань повинна бути створена кнопка генерації даних. В полях в тренувальній частині потрібно зробити підказки (підсвічування правильних та неправильних відповідей), щоб користувач бачив в чому він помилився і зміг ще раз обдумати, а після відредагувати відповідь.

Підказку або ж підсвічування можна прив'язати до кнопки перевірки, щоб відповіді міняли колір після оцінки, а не під час проходження завдань. Також додати можливість редагування відповідей буде влучним рішенням. В такому випадку користувач зможе відредагувати невірну відповідь і після цього переконатись чи відповіді були виправлені коректно. Також потрібно зробити так, щоб цей процес можна повторювати декілька або ж скільки завгодно разів з тими ж самими даними, до тих пір, поки результат не задовольнить користувача.

Кнопці генерації даних окрім функції створення даних, можна присвоїти оновлення попередніх відповідей, таким чином користувачу не потрібно буде власноруч видаляти попередні відповіді, що хоч і незначно, але економить час. Зважаючи на завдання з ір-адресації і параметри, що можливо знайти, вдалим рішенням було б додати функцію вимкнення полів відповідей. Це збільшить варіативність завдань в рази, не потрібно буде для кожного набору параметрів для пошуку створювати окремі завдання, оскільки вимкнені поля не оцінюються. Таким чином, всі варіанти завдань можуть відбуватися в одному

вікні. Також, це дозволить зосередитись на проблемних моментах, в яких користувач не повністю розуміється.

Таким чином, ця програма є унікальною. Можливість подальшого розвитку і додання нових тем та розділів робить потенціал «IPTeacher» практично безмежним.

1.5.2 Постановка завдання на розробку модулю контролю знань.

«IPTester» – це модуль для контролю знань, для навчально-тренувальної програми.

Програма «IPTester» подібна за функціоналом до «IPTeacher». В ній доречно буде реалізовати ті ж завдання, але з спираючись на особливості модуля контролю. Структура завдань ідентична «IPTeacher», проте відповіді після оцінки коректності редагувати не можна буде. Це означає що у користувача буде всього лиш одна спроба на проходження завдання із згенерованими даними, що виключає підбір правильної відповіді, як у «IPTeacher».

В модулі контролю знань абсолютно недоречний теоретичний матеріал, адже користувач в такому випадку зможе підглядати. В такому випадку модуль контролю втрачає сенс. Нова генерація, як і в «IPTeacher», повинна оновлювати попередні відповіді. Підсвічування відповідей в модулі не є потрібним, планується замінити його функцією оцінювання у відсотках. Кнопка оцінки виводитиме бал, який показуватиме, наскільки успішно користувач впорався із завданням. Кожне поле оцінюватиметься в однакову кількість відсотків в залежності від кількості полей для відповіді. Також, в модуль контролю потрібно буде додати таймер, що виводитиме час, який виділяється на проходження завдання. Після закінчення часу оцінка відбуватиметься автоматично.

В підсумку, в модулі контролю знань для навчально-тренувальної програми передбачений функціонал, який дозволить проводити оцінювання чесно.

Висновки з першого розділу.

При виконанні даного розділу кваліфікаційної робити був проведений аналіз навчальних програм в цілому. Проведено аналіз актуальності сайтів та комп’ютрних програм навчального призначення. Проведено аналіз вимог та особливостей подібних програм на існуючих прикладах. Проведений аналіз розробки програмного забезпечення з урахуванням стандартів. Було розглянуто мову програмування C# та вибрано найоптимальніше середовище розробки для неї. Також було проаналізовано поставлене завдання на розробку, поставлений план виконання проєкту спираючись на попередні пункти.

РОЗДІЛ 2. РОЗРОБКА СТРУКТУРИ, ВИБІР ІНСТРУМЕНТІВ ДЛЯ C# ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

2.1. Розробка структури програми.

2.1.1. Розробка структури модулю контролю знань «IPTeaster».

Структура запуску доволі проста. Файл запуску відкриє програму і перед користувачем буде вибір, який складається з 4 вкладок.

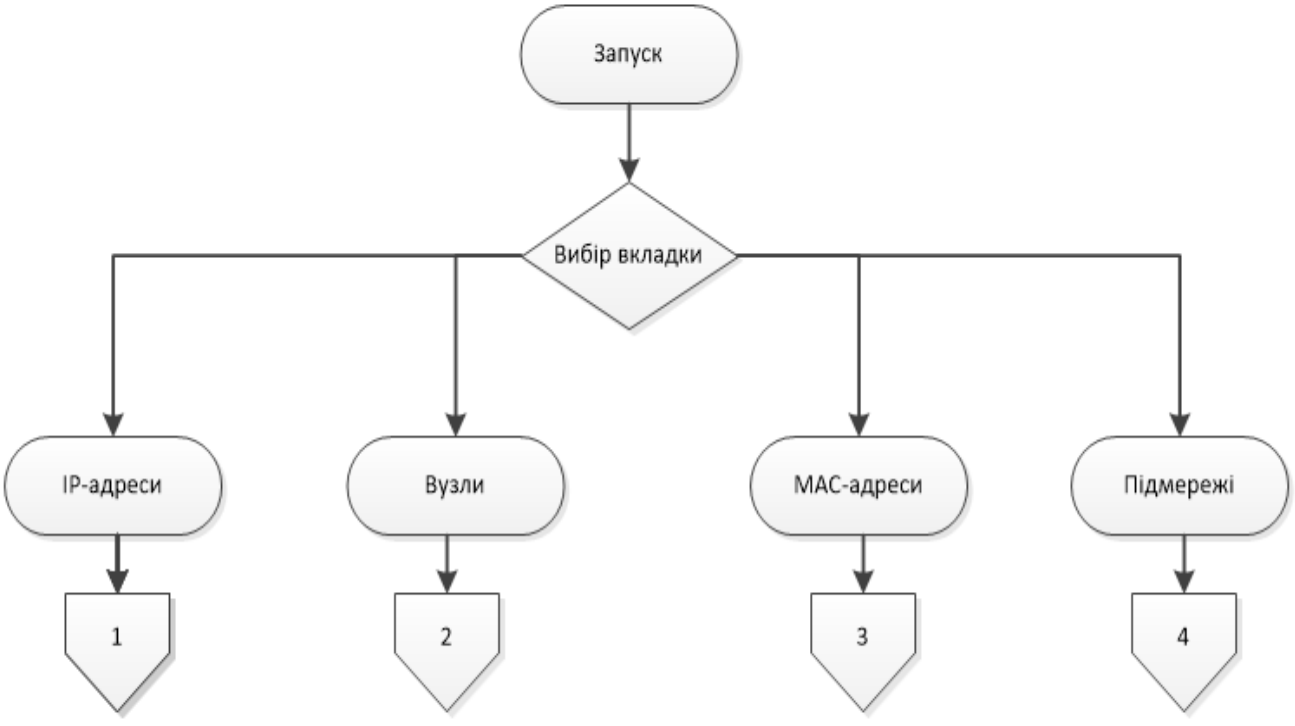


Рис.2.1. Блок-схема запуску програми

Для запуску та функціонування програми перш за все потрібен .NET Framework. Сам файл запуску (.exe) програми знаходиться в папці bin і виглядає наступним чином:



Рис.2.2. Файл запуску програми

Файли двох програми мають однакову назву. Це через те, що розробка функціоналу розпочиналась з модулю контролю, а далі на його основі була створена навчально-тренувальна програма. Оскільки, функціонал обрахунку і деяких кнопок ідентичні, було вирішено не змінювати назви, щоб зекономити час розробки.

Дані всіх завдань, окрім вкладки «Підмережі», генеруються за допомогою генератора випадкових чисел. У вкладці «Підмережі» вирішено створити варіанти для завдань у базі даних.

Структура програми передбачає 4 теми, розділені по вкладкам:

– IP-мережі;

– Вузли;

– MAC-адреси;

– Підмережі.

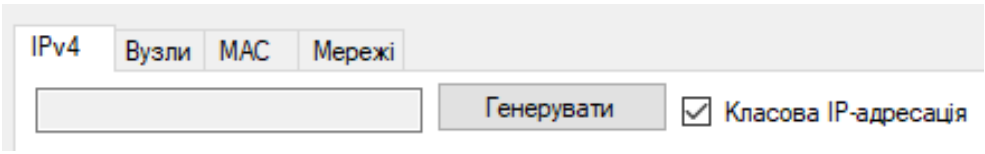


Рис.2.3. Вигляд вкладок реалізованих в програмі

У вкладках «IP-мережі», так само, як і у вкладці «Вузли», реалізований вибір класового або безкласового підходу в одній вкладці. Він відбувається шляхом переключення відповідного поля. Таке рішення значно оптимізує функціонал програми, адже не потребує створення окремих вкладок в програмі для кожного з підходів.

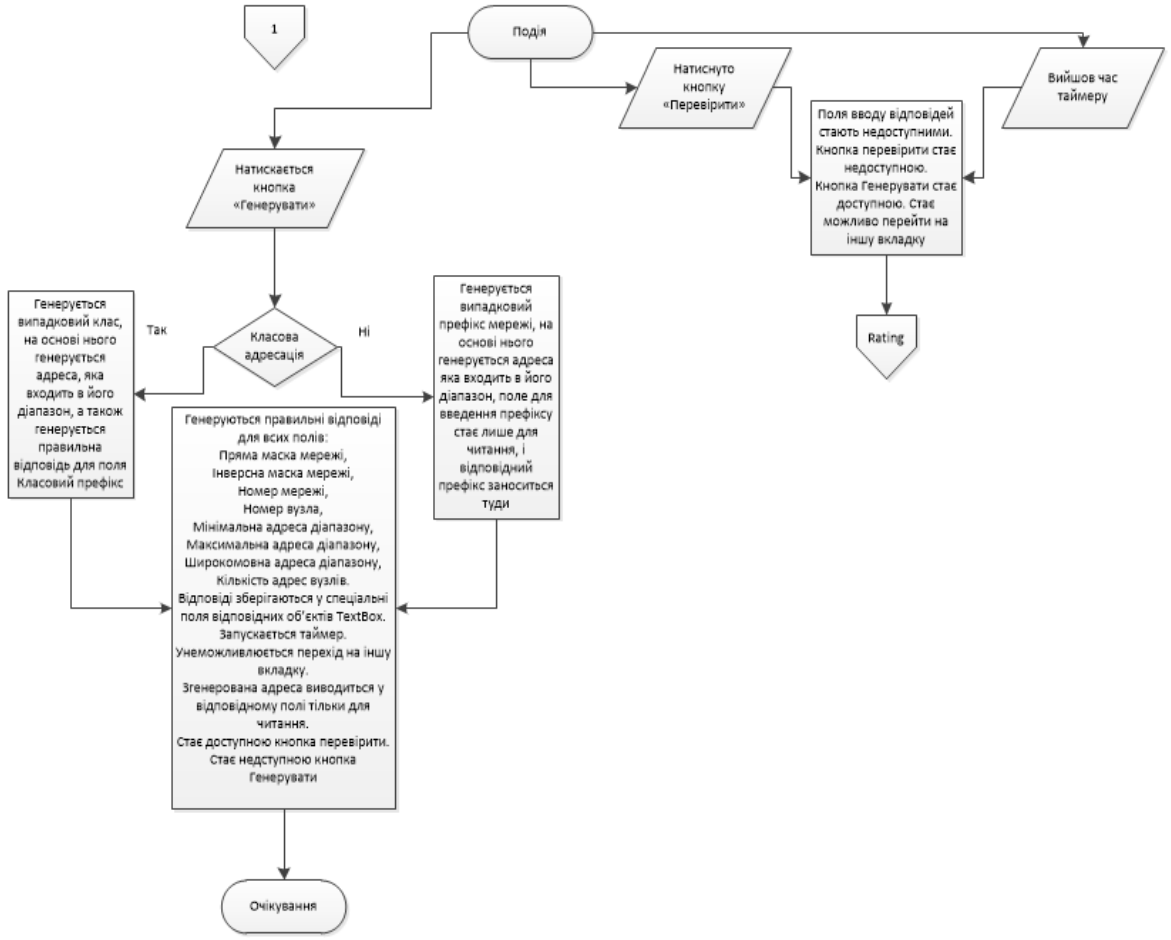


Рис.2.4. Блок-схема вкладки «IP-адреси»

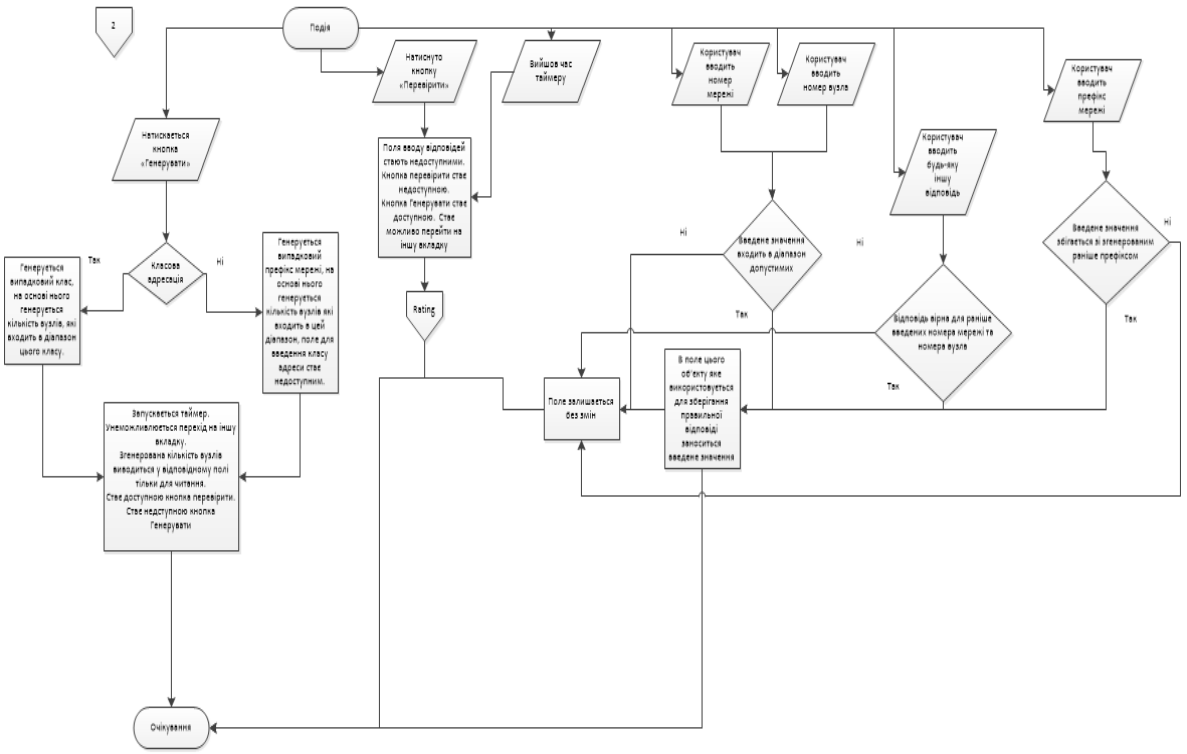


Рис.2.5. Блок-схема вкладки «Вузли»

Завдання з MAC-адресами виконане у вигляді невеличкого тесту з вибором правильних відповідей.

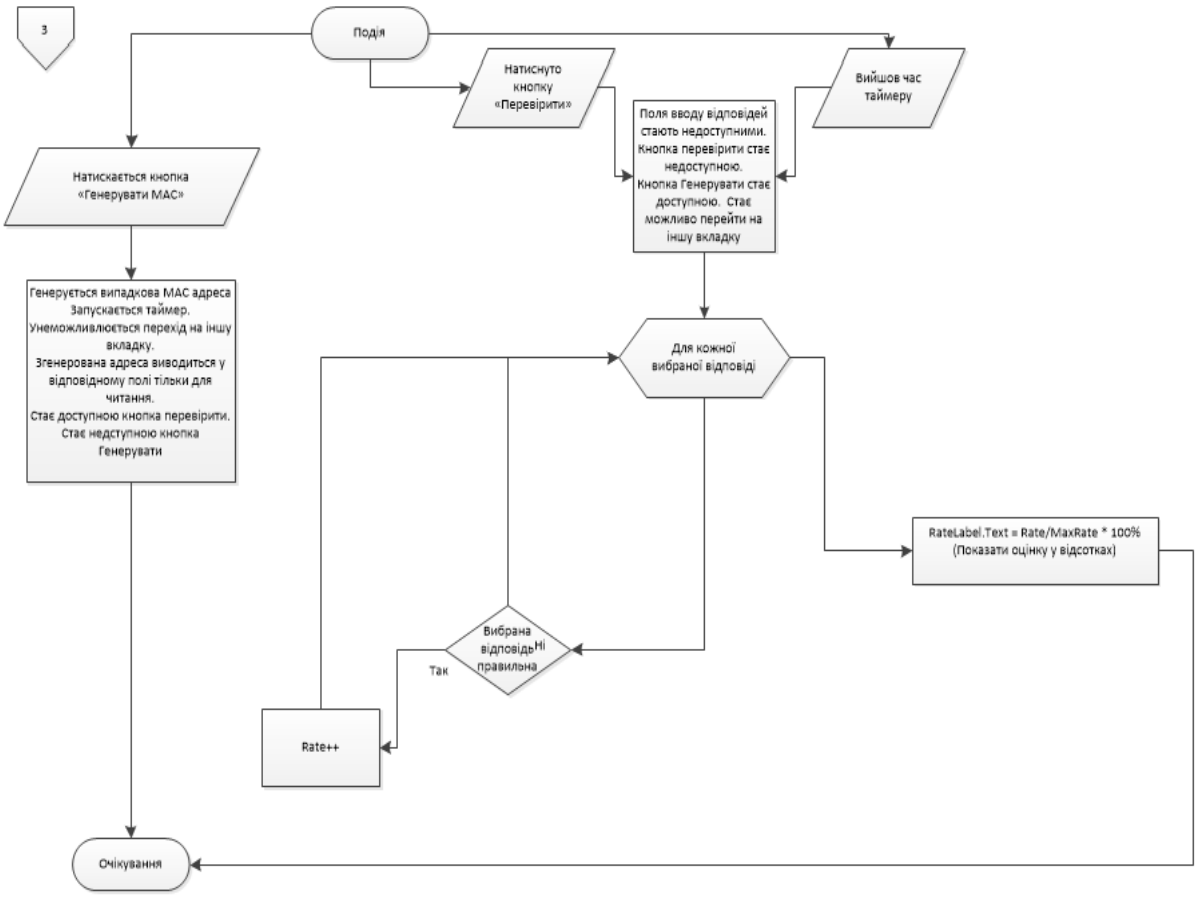


Рис.2.6. Блок-схема вкладки «MAC-адреси»

Вкладка «Підмережі» побудована з деякими відмінностями від попередніх. Структура цієї вкладки передбачає 8 завдань, розділених на блоки, проте оцінка обраховує відповіді за всі завдання разом. В кожну таблицю по бажанню можна додати необмежену кількість різних варіантів завдань. При цьому функціональні зміни в програмі не потрібні.

Завдання для вкладки «Підмережі» підбираються з бази даних SQLite (файл subnet.db3).



Рис.2.7. Файл БД

В базі даних в кожному завданні зберігаються тільки варіанти даних, а розрахунок правильних відповідей програма виконує самостійно. Це означає,

що додавання нових варіантів даних, їх зміна або ж видалення не спричинить помилок в розрахунках при такому ж функціоналі.

Для кожного завдання із теми «Підмережі» створений окремий клас в окремому файлі .cs.

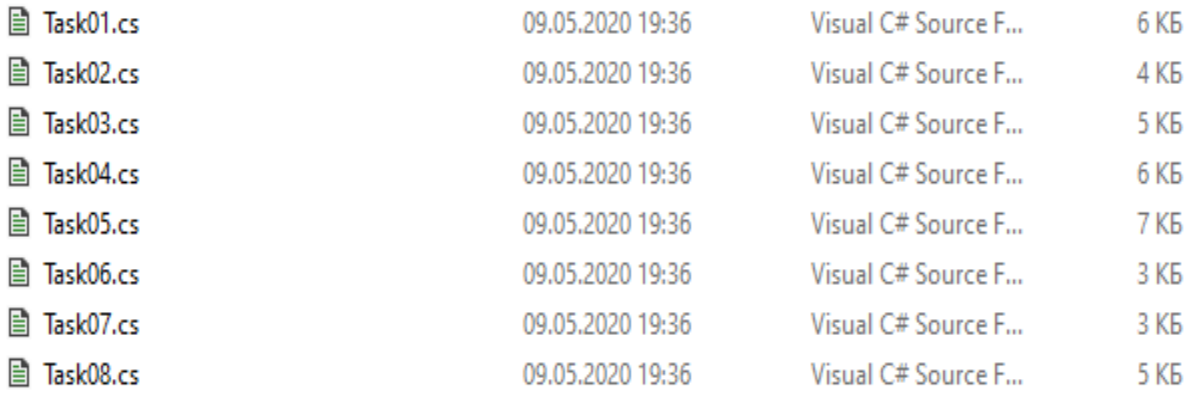


Рис.2.8. Класи вкладки «Підмережі»

Таке рішення забезпечує можливість швидкої заміни функціоналу, а також дозволяє скористатись методом повторного використання коду. Це методологія проектування комп'ютерних та інших систем, яка полягає в тому, що система частково або повністю повинна складатись із частин, написаних раніше компонентів і/або частин іншої системи, а ті компоненти повинні застосовуватись більше одного разу).

База даних SQLite складається із наступних таблиць:

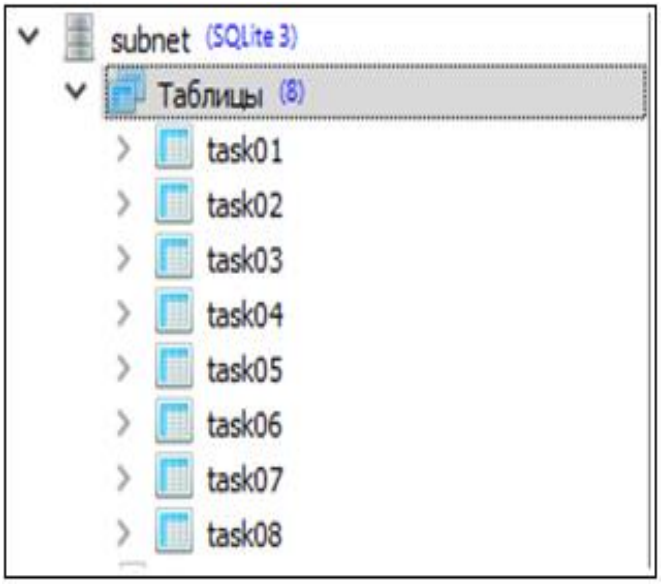


Рис.2.9. Таблиці БД SQLite

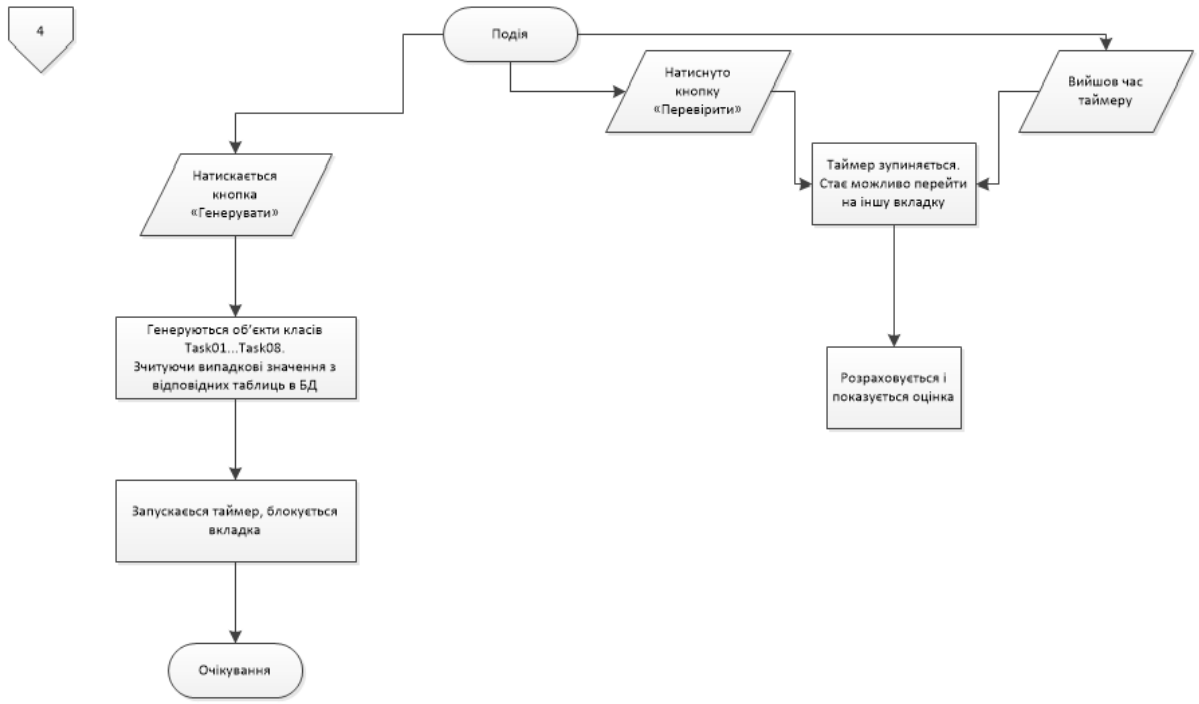


Рис.2.10. Блок-схема вкладки «Підмережі»

На Рис. 2.11. показано методи для класів Task01, Task02, Task03, …, Task08. Вони виступають окремими підпрограмами.

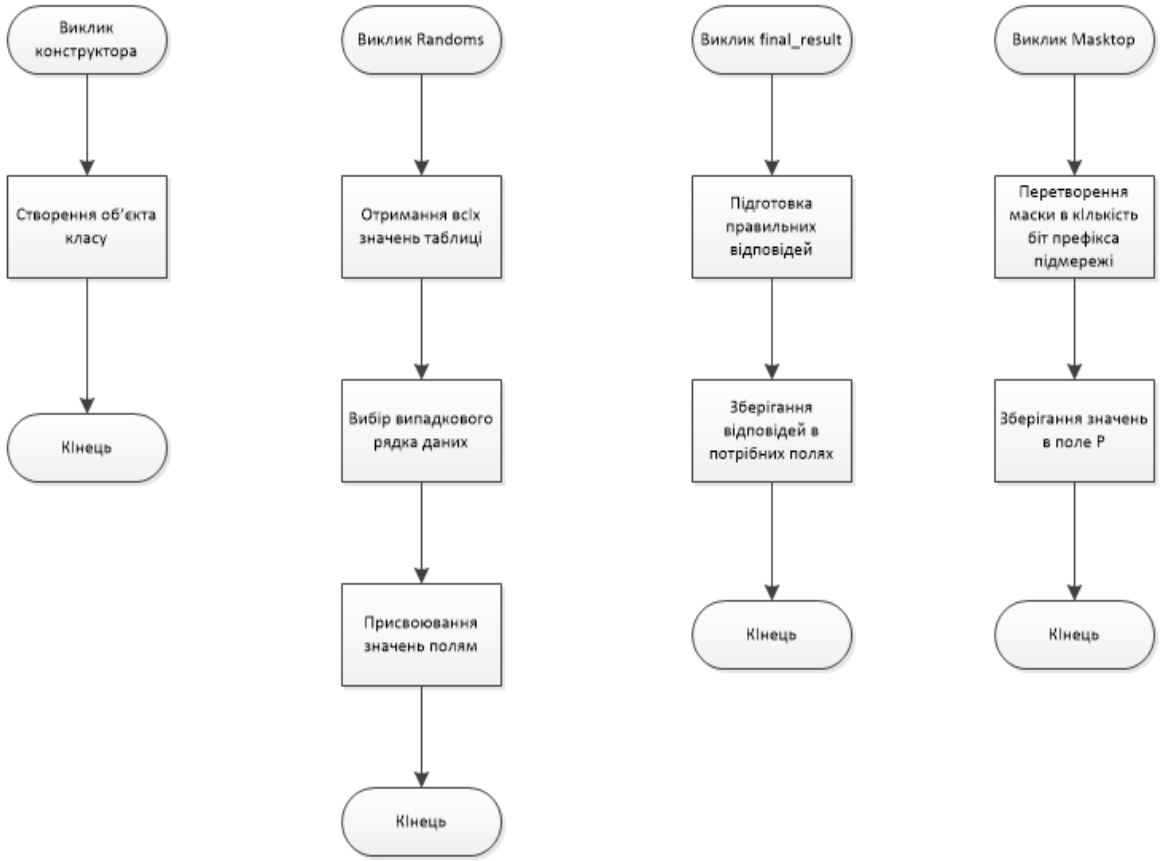


Рис.2.11. Блок-схема класів Task01, Task02, Task03, …, Task08

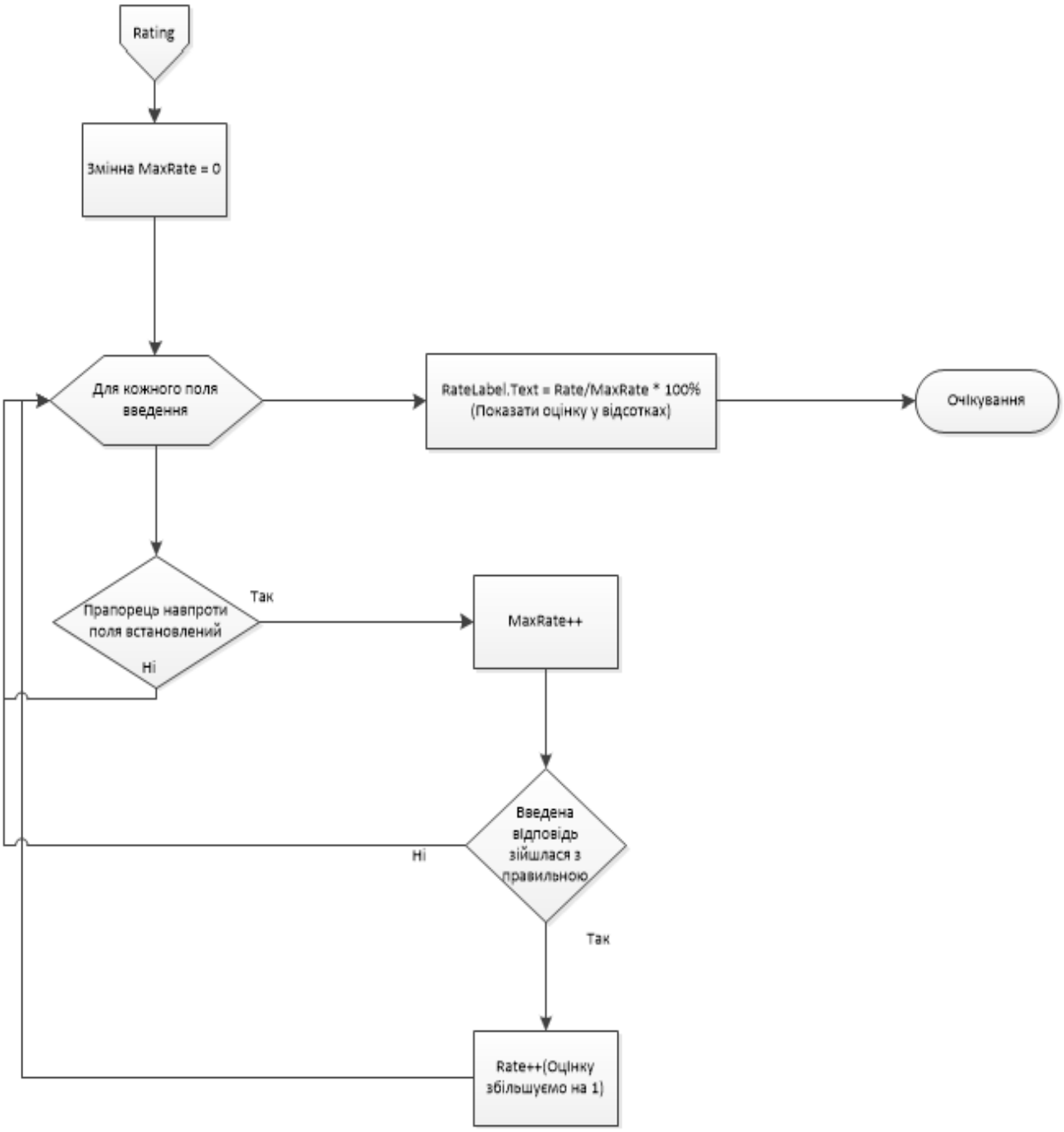


Рис.2.12. Блок-схема оцінки

2.1.2. Розробка структури навчально-тренувальної програми «IPTeacher».

«IPTeacher», як вже відомо, розроблявся на основі «IPTester», тож структури двох програм дуже подібні. Алгоритм обрахунку залишились без змін, проте змінився функціонал деяких кнопок та полів та самі елементи в цілому, спираючись на особливості навчально-тренувальної програми. «IPTeacher» він містить практично ті самі файли що і в модулі контролю знань.

Відмінними файлами від модулю контрою знань у навчально-тренувальній програмі є html файли. Вони заходяться у тій ж папці що і запускний файл .exe.

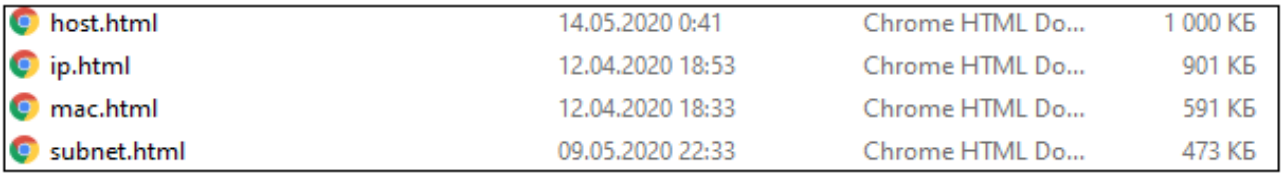


Рис.2.13. Файли html

За допомогою цих файлів здійснюється вивід інформації для кожної вкладки програми. Наразі в цих файлах знаходиться текст методичних рекомендацій щодо вирішення завдань реалізованих в програмі. Проте, їх можна легко редагувати за допомогою будь-якого текстового редактору або ж навіть повністю замінити на інший. Для редагування змісту найкраще підійде текстовий редактор Microsoft Word. В ньому найзручніше відредагувати файл, щоб він добре вписувався в ширину рамок поля для файлу читання.

Файл читання для кожної теми повинен відображатися одразу при відкритті програми, при виборі нової вкладки мінятись на відповідний темі.

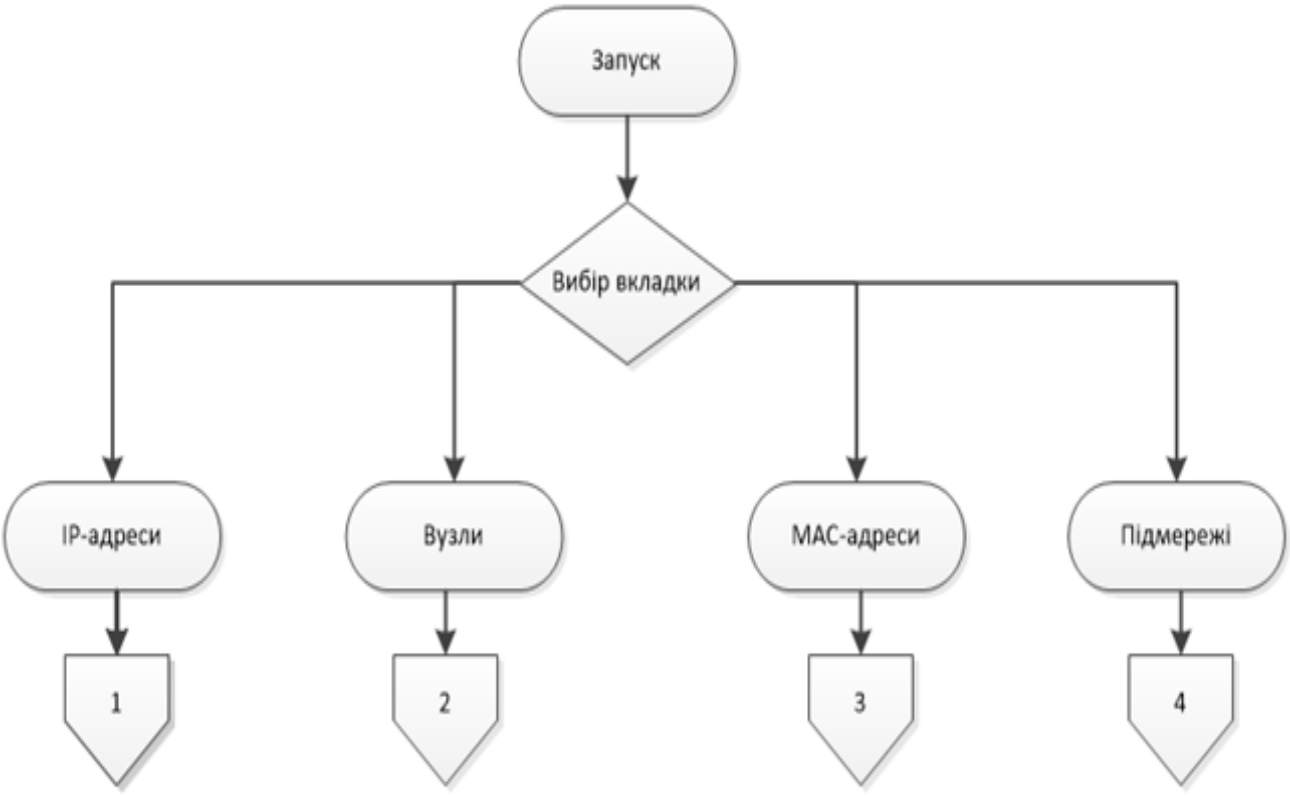


Рис.2.14. Блок-схема запуску програми

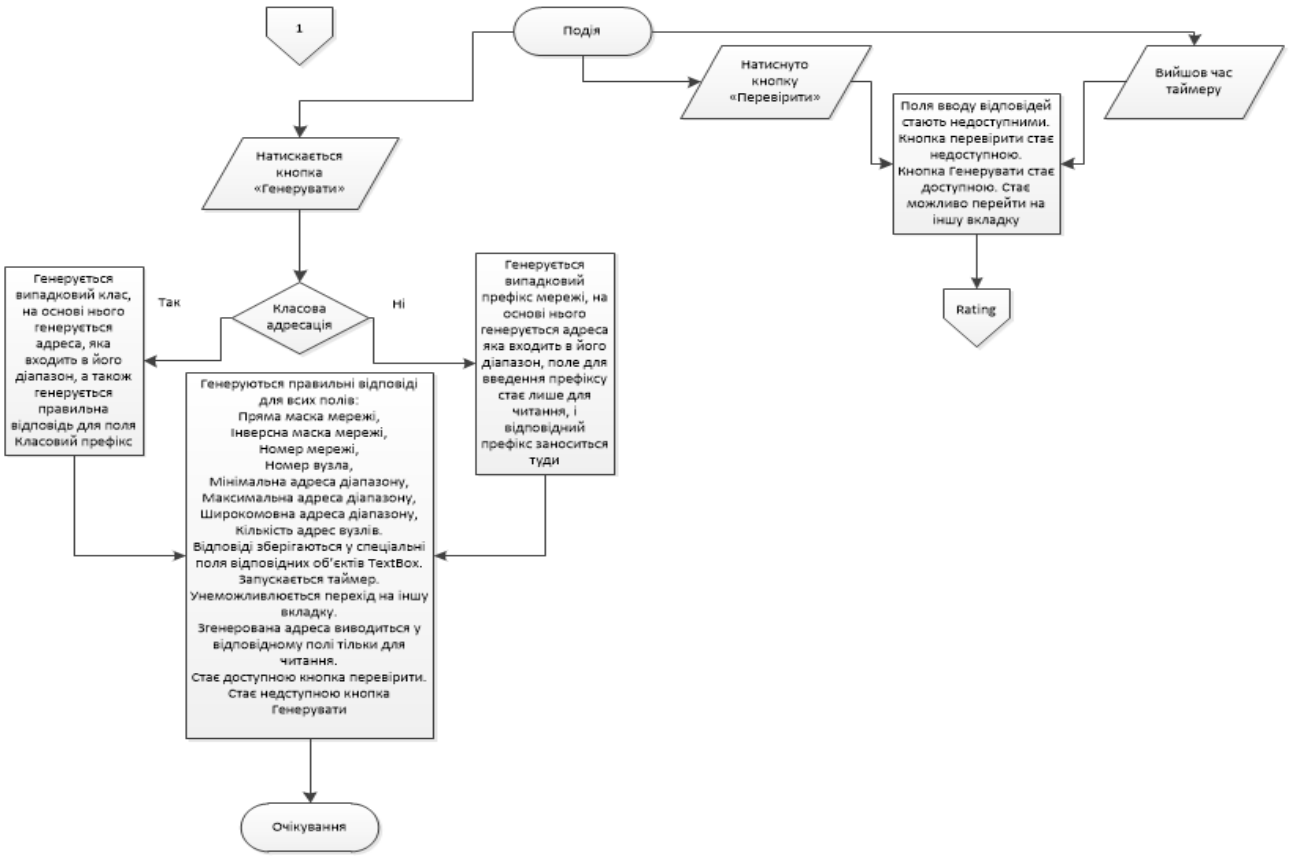


Рис.2.15. Блок-схема вкладки «IP-адреси»

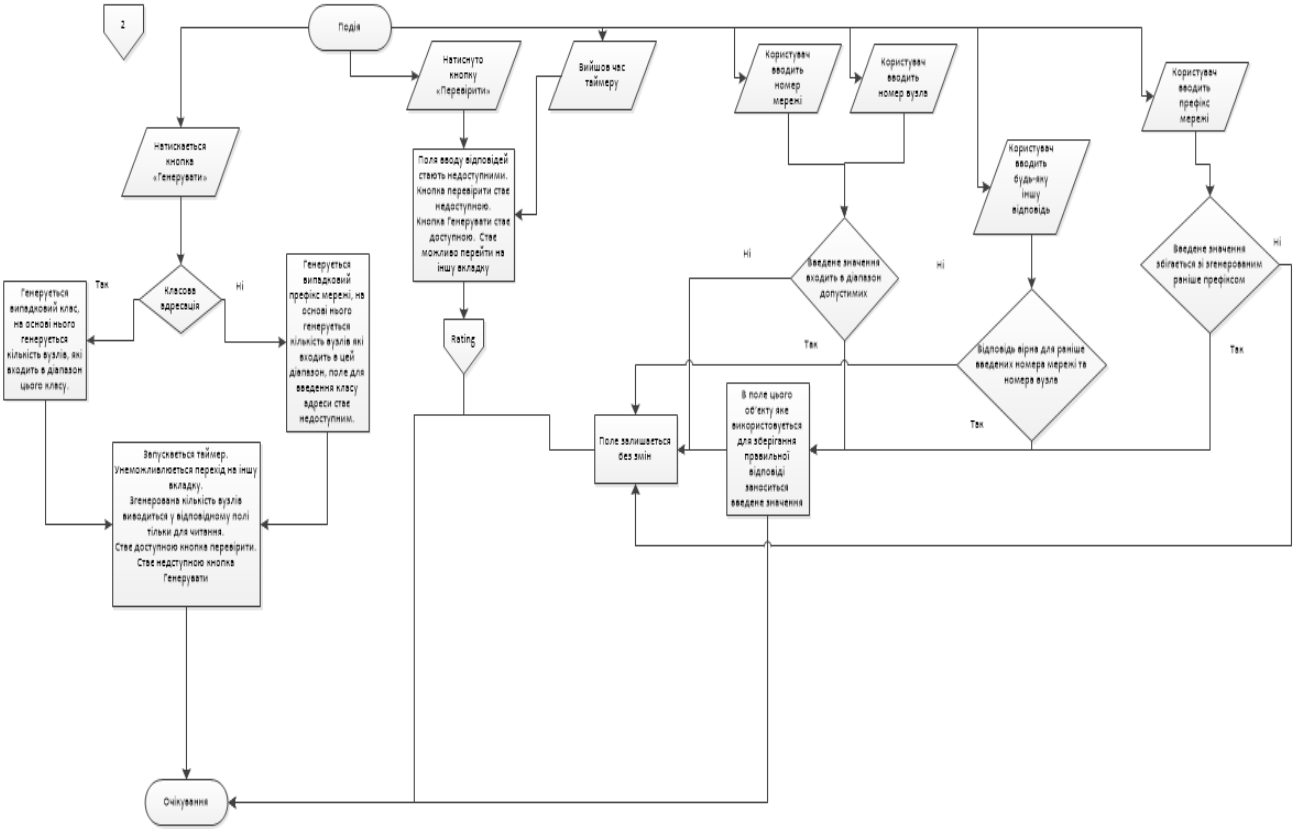


Рис.2.16. Блок-схема вкладки «Вузли»

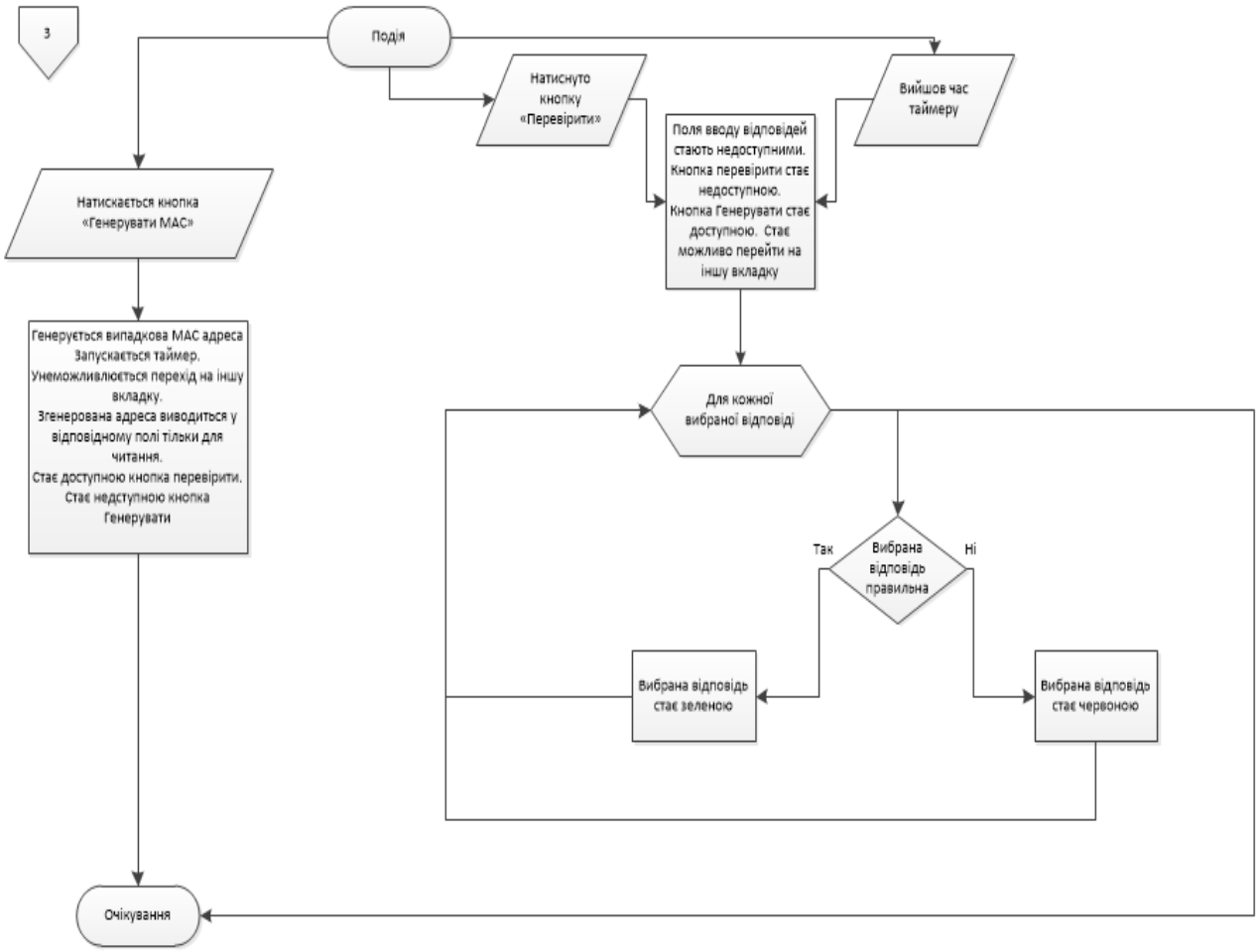


Рис.2.17. Блок-схема вкладки «MAC-адреси»

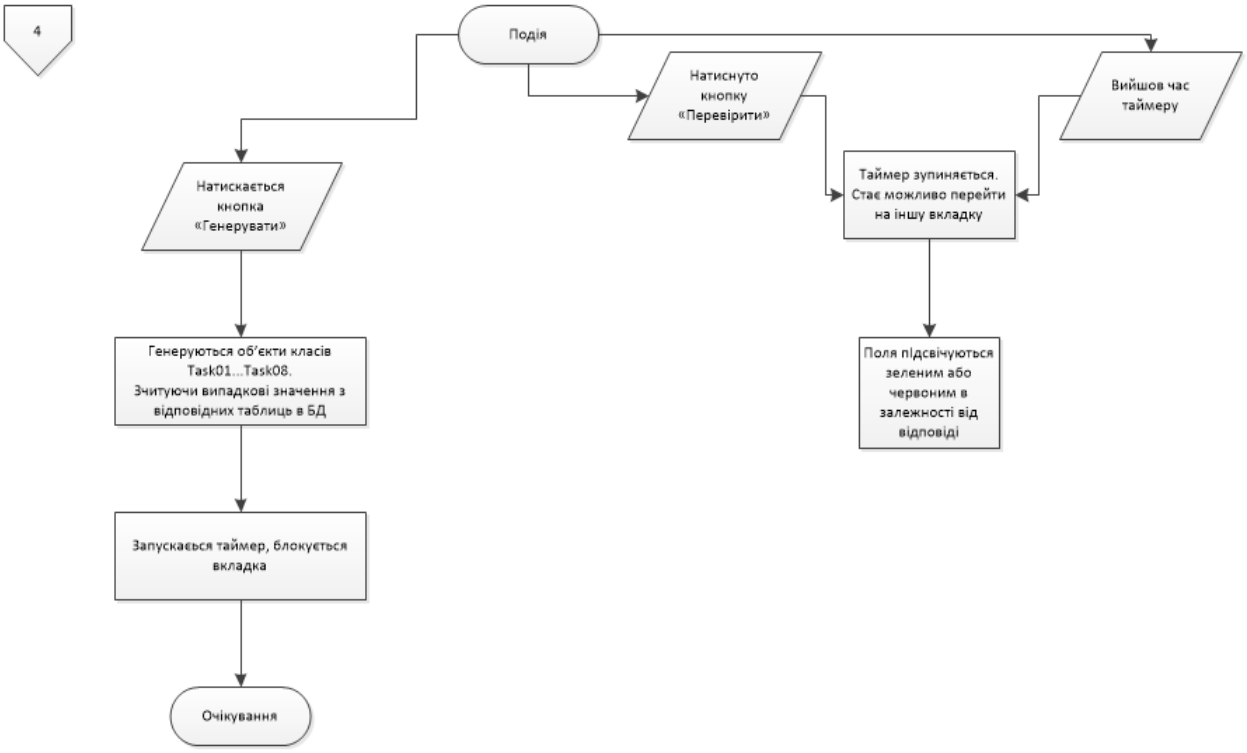


Рис.2.18. Блок-схема вкладки «Підмережі»

Методи для класів Task01, Task02, Task03, …, Task08 виступають окремими підпрограмами, тож для них була створена окрема блок-схема.

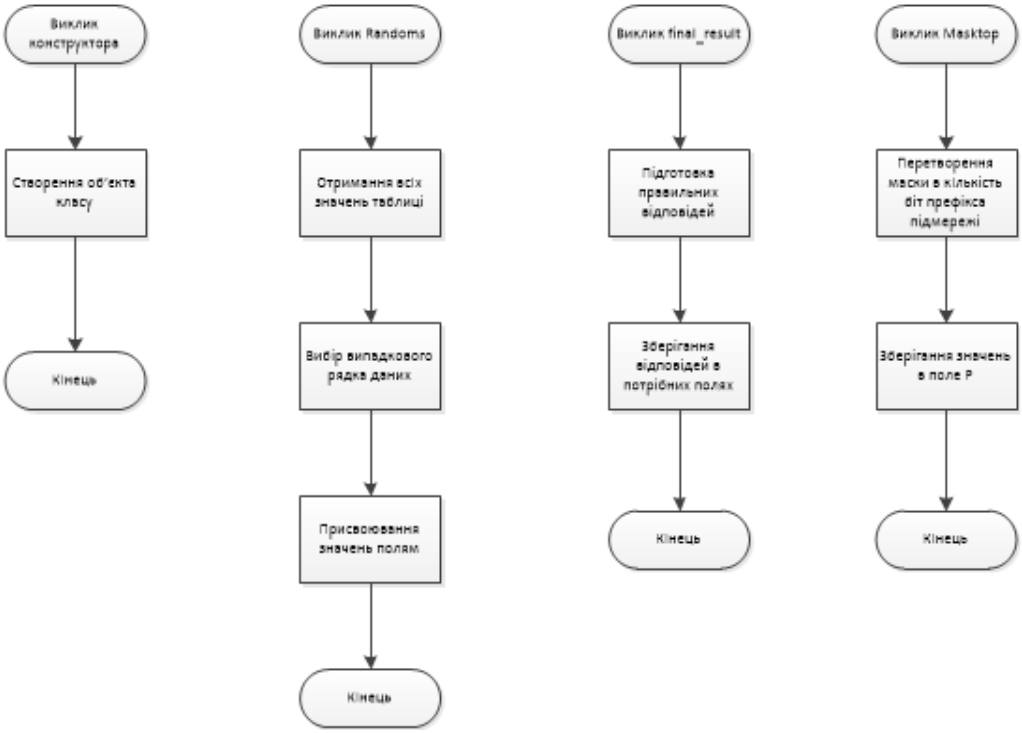


Рис.2.19. Блок-схема класів Task01, Task02, Task03, …, Task08

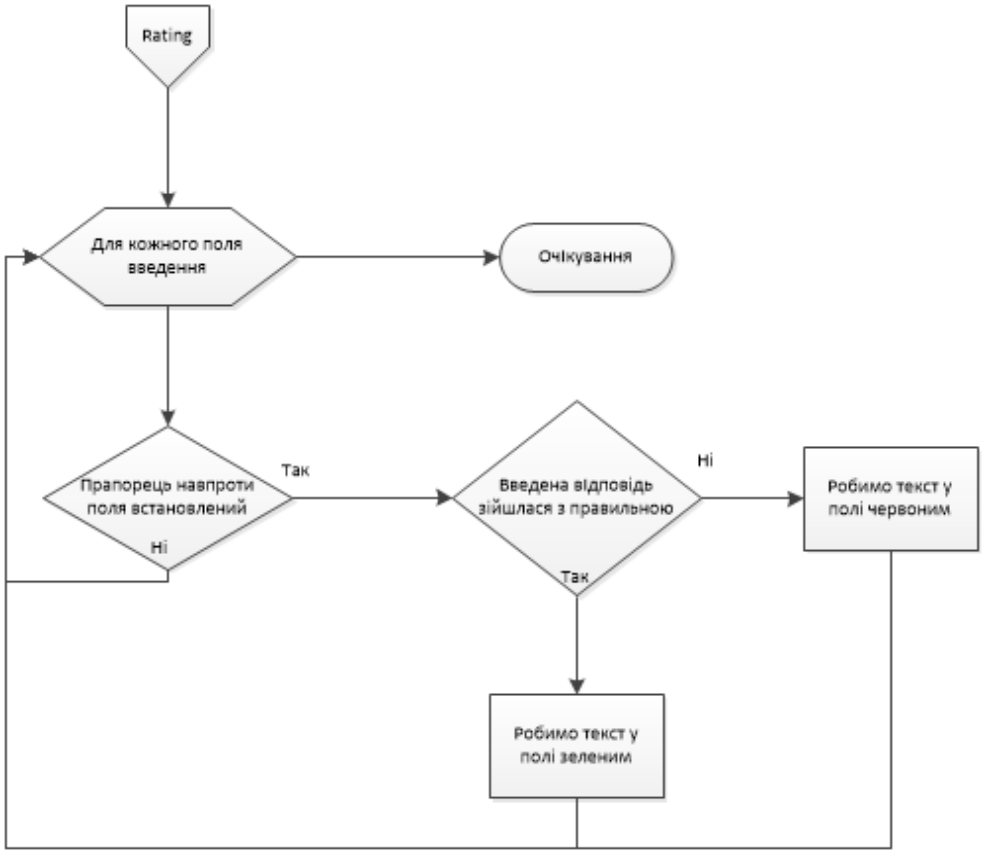


Рис.2.20. Блох-схема оцінки

2.2. Вибір інструментів C# для реалізації навчально-тренувальної програми.

Для реалізації програми був обраний інтерфейс Microsoft Windows Forms. Завдяки простоті редагування і можливості запуску на будь-якій Windows-based системі (за умови наявності відповідної версії .NET Framework) це рішення є ідеальним. Також програму, зроблену за допомогою Windows Forms можна запустити на Linux-based системах за умови наявності графічного інтерфейсу (Gnome, XFCE, ін.). Завдання з полями знаходиться в блоці groupBox. GroupBox – це спеціальний контейнер, який обмежений від решти форми. Він має заголовок, який встановлюється через властивість Text. Безпосередньо, текст завдання знаходиться в елементі Windows Forms Label. Елемент управління Label призначений для створення підписів до інших елементів управління або для виведення інформаційних повідомлень прямо на поверхні форми. Текст методичних вказівок для виконання лабораторної роботи виводиться за допомогою елемента WebBrowser.

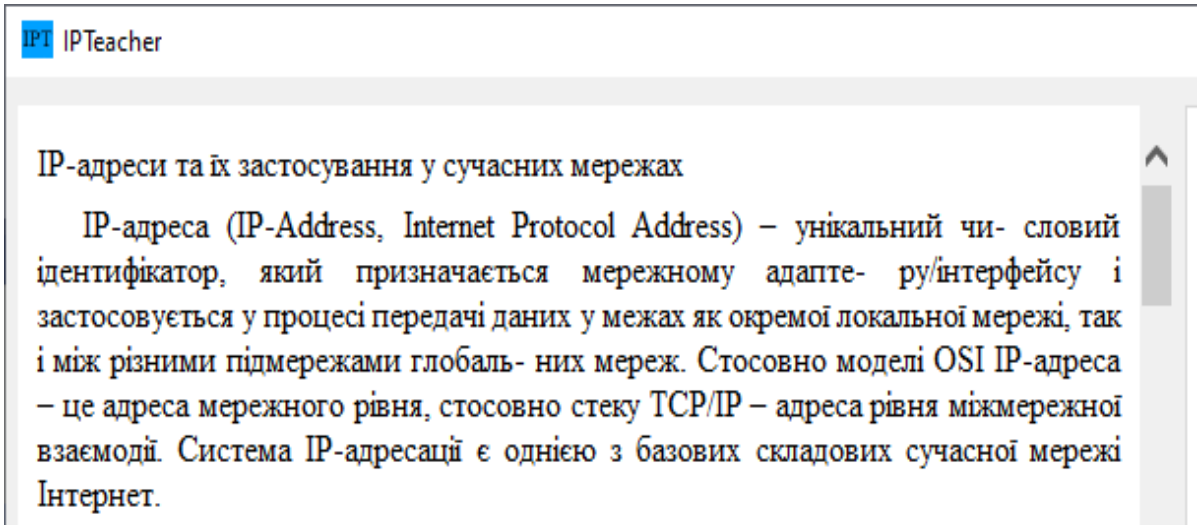


Рис.2.21. Вивід тексту методичних вказівок за допомогою елемента WebBrowser

WebBrowser надає функції інтернет-браузера, дозволяючи завантажувати і відображати контент з мережі Інтернет, а також дозволяє відображати локальні файли з форматуванням і вмістом HTML – файлів (HyperText Markup Language - стандартизовану мову розмітки документів у Всесвітній павутині).

Незважаючи на досить обмежений функціонал (даний елемент не є повноцінним веб-браузером, і можливості по його налаштуванні і зміні досить обмежені), цього достатньо для зручного відображення методичних вказівок. Для введення відповідей буде використовуватися елемент TextBox. Даний елемент дозволяє зчитувати строкові дані, що вводяться користувачем, а також в разі показувати користувачеві, чи була відповідь правильна, завдяки зміні кольору тексту.

Елементи GroupBox в програмі здебільшого використовуються як оформлення, щоб відділити елемент один від одного візуально. Проте, у вкладці «Підмережі» вони мають важливий функціонал. В кожен GroupBox внесене завдання і до кожного з них прив'язаний елемент CheckBox, який його вимикає, так само, як поля у вкладках «IP-адреси» та «Вузли».

2.3. Розробка програмного забезпечення.

Розробка програмного забезпечення відбувалась з модуля для контролю знань, а навчально-тренувальна програма була модифікована спираючись на її ціль та особливості. Проте, основи «IPTeacher» і «IPTester» однакові, через те що вкладки, завдання та алгоритм обрахунку ідентичні і не потребують змін.

Була створена необхідна база даних з варіантами відповідей за допомогою SQLite Studio.

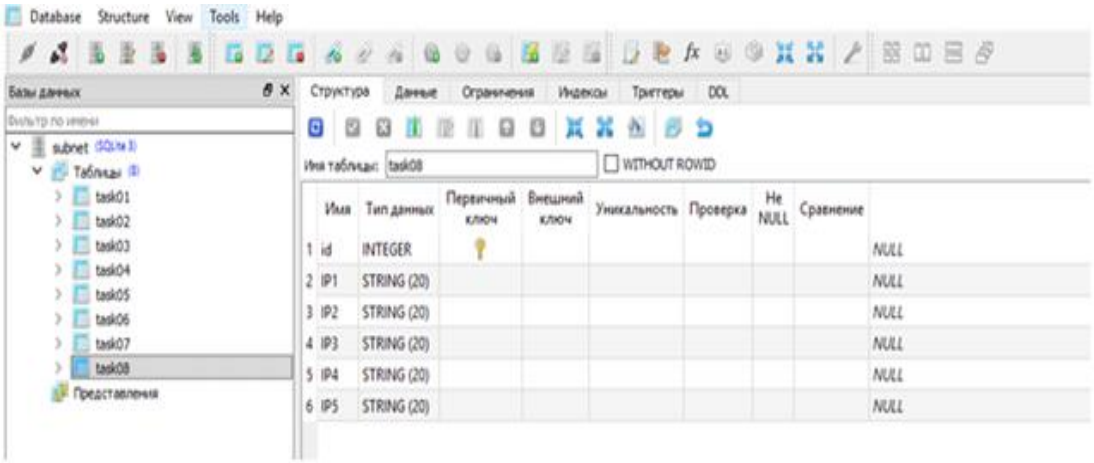
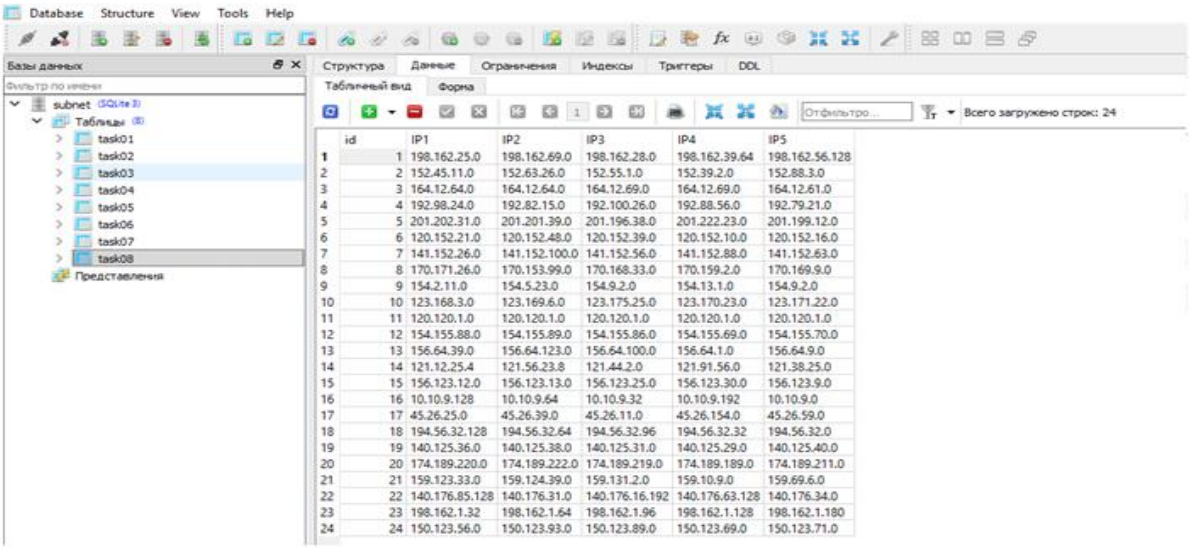


Рис.2.22. База даних SQLite з варіантами відповідей

Також таблиці в базі даних були заповнені варіантами завдань:



	id	IP1	IP2	IP3	IP4	IP5
1	1	198.162.25.0	198.162.69.0	198.162.28.0	198.162.39.64	198.162.56.128
2	2	152.45.11.0	152.63.26.0	152.55.1.0	152.39.2.0	152.88.3.0
3	3	164.12.64.0	164.12.64.0	164.12.69.0	164.12.69.0	164.12.61.0
4	4	192.98.24.0	192.82.15.0	192.100.26.0	192.88.56.0	192.79.21.0
5	5	201.202.31.0	201.201.39.0	201.196.38.0	201.222.23.0	201.199.12.0
6	6	120.152.21.0	120.152.48.0	120.152.39.0	120.152.10.0	120.152.16.0
7	7	141.152.26.0	141.152.100.0	141.152.56.0	141.152.88.0	141.152.63.0
8	8	170.171.26.0	170.153.99.0	170.168.33.0	170.159.2.0	170.169.9.0
9	9	154.2.11.0	154.5.23.0	154.9.2.0	154.13.1.0	154.9.2.0
10	10	123.168.3.0	123.169.6.0	123.175.25.0	123.170.23.0	123.171.22.0
11	11	120.120.1.0	120.120.1.0	120.120.1.0	120.120.1.0	120.120.1.0
12	12	154.155.88.0	154.155.89.0	154.155.86.0	154.155.69.0	154.155.70.0
13	13	156.64.39.0	156.64.123.0	156.64.100.0	156.64.1.0	156.64.9.0
14	14	121.12.25.4	121.56.23.8	121.44.2.0	121.91.56.0	121.38.25.0
15	15	156.123.12.0	156.123.13.0	156.123.25.0	156.123.30.0	156.123.9.0
16	16	10.10.9.128	10.10.9.64	10.10.9.32	10.10.9.192	10.10.9.0
17	17	45.26.25.0	45.26.39.0	45.26.11.0	45.26.154.0	45.26.59.0
18	18	194.56.32.128	194.56.32.64	194.56.32.96	194.56.32.32	194.56.32.0
19	19	140.125.36.0	140.125.38.0	140.125.31.0	140.125.29.0	140.125.40.0
20	20	174.189.220.0	174.189.222.0	174.189.219.0	174.189.189.0	174.189.211.0
21	21	159.123.33.0	159.124.39.0	159.131.2.0	159.10.9.0	159.69.6.0
22	22	140.176.85.128	140.176.31.0	140.176.16.192	140.176.63.128	140.176.34.0
23	23	198.162.1.32	198.162.1.64	198.162.1.96	198.162.1.128	198.162.1.180
24	24	150.123.56.0	150.123.93.0	150.123.89.0	150.123.69.0	150.123.71.0

Рис.2.23. База даних SQLite з варіантами завдань

Були створені відповідні класи для вкладки «Підмережі» і додані в проєкт рішення.

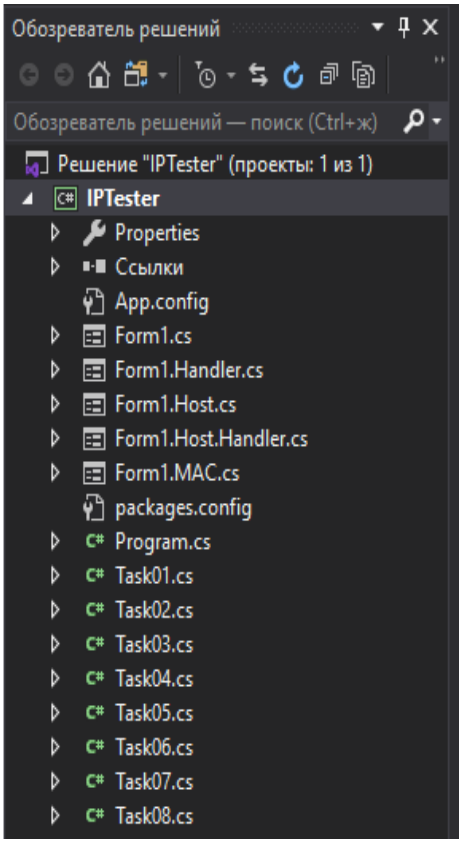


Рис.2.24. Класи вкладки «Підмережі»

Так як в програмі потрібно використовувати роботу з базами даних SQLite, через менеджер пакетів NuGet був встановлений і доданий пакет

System.Data.SQLite. Він включає в себе наступні модулі: System.Data.SQLite.Core, System.Data.SQLite.EF6, System.Data.SQLite.Linq.

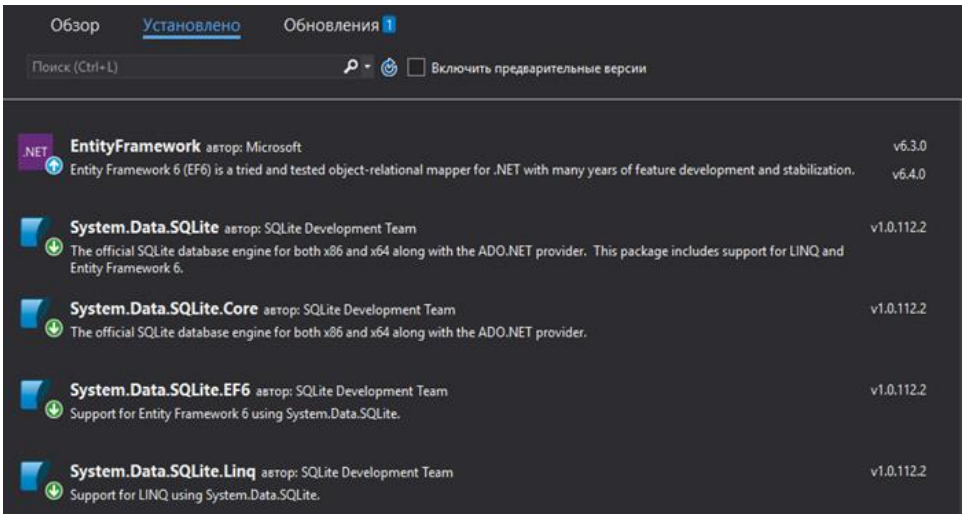


Рис.2.25. Встановлені модулі для роботи з SQLite

Ці модулі дозволяють повноцінно взаємодіяти з СУБД SQLite. Менеджер пакетів NuGet автоматично додає необхідні бібліотеки в папки Debug / Release для успішного налагодження і роботи програми.

Після виконання даних підготовчих робіт будується графічний інтерфейс для взаємодії користувача з програмою. В процесі роботи додаються елементи з панелі елементів середовища розробки Visual Studio:

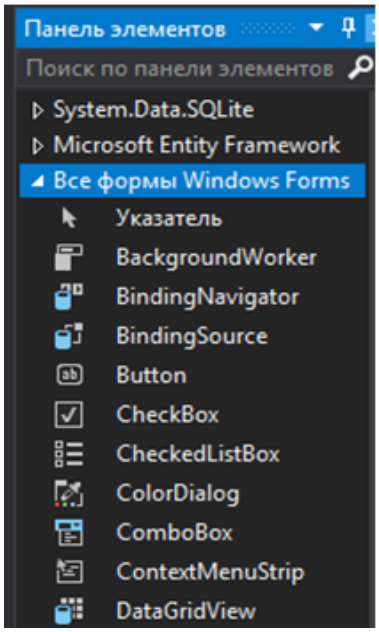


Рис.2.26. Панель елементів Visual Studio

Після доповнення елементів через меню «Властивості» змінюються методи і параметри графічних елементів, прив’язуються події:

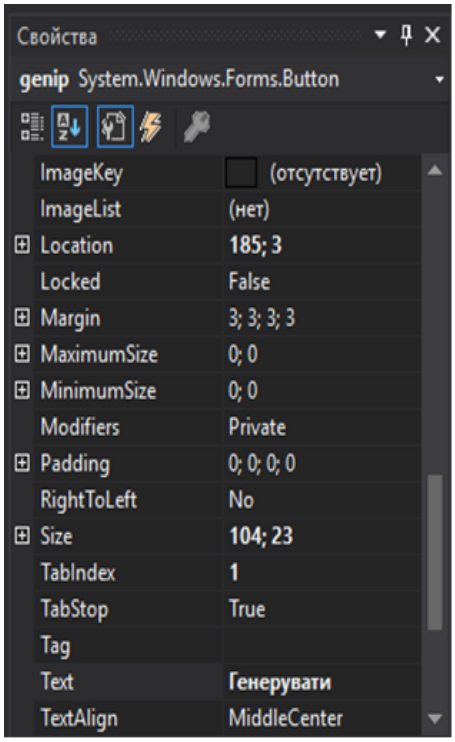


Рис.2.27. Зміна методів і параметрів графічних елементів та прив’язка подій

Після виконання даних дій програма проходить тестування і відбувається коригування (перевірка коректності даних, що вводяться, коректне відображення елементів при відповідній ширині / висоті відображення вікна програми і т. д.). Після цього можна побачити результат:

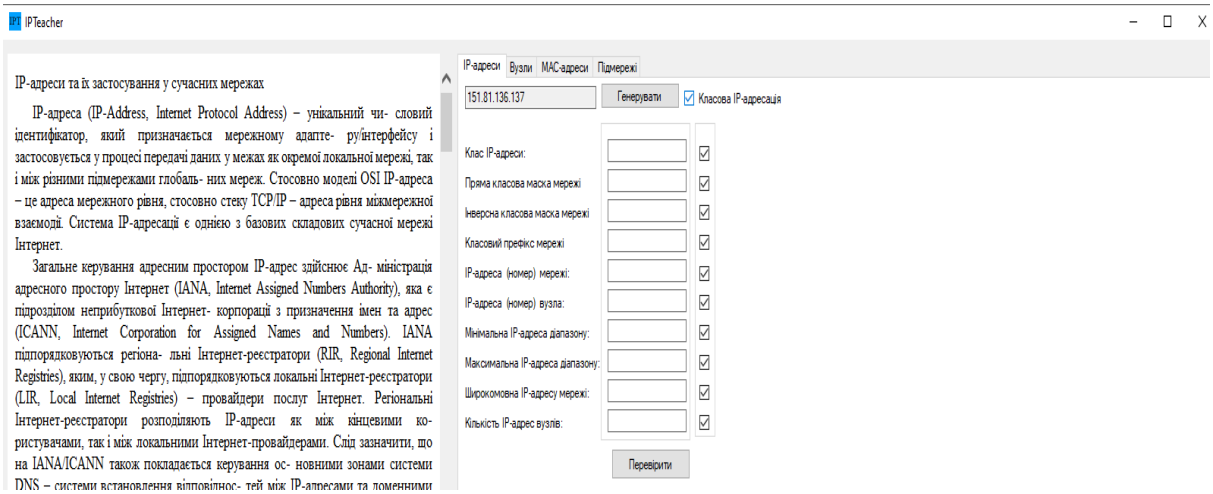


Рис.2.28. Результат зміни параметрів елементів

Висновки з другого розділу

В другому розділі було розглянуто структуру навчально-тренувальної програми: її основних файлів та алгоритм роботи запуску програми, кожної вкладки та оцінки. Також, були розглянуті вибрані інструменти для реалізації навчально-тренувальної програми та модулю для контролю знань. Була розглянута тема розробки програмного забезпечення, необхідні додаткові програми та ресурси та тонкості розробки.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ ПРАЦЕЗДАТНОСТІ ТА ІНСТРУКЦІЯ КОРИСТУВАННЯ ПРОГРАМОЮ

3.1. Дослідження працездатності та ефективності

3.1.1 Дослідження модулю контролю знань.

В «IPTester» було проведено ряд перевірок на коректність генерації даних, обрахунку завдань кожної теми реалізованих в програмі, коректного спрацювання всіх активних елементів у програмі та роботи програми в цілому. Перевірка відбувалась в декілька етапів, тому що виявлялись помилки та баги, які потрібно було усунути. Перевірка обрахунку одного конкретного поля, була перевірена велика кількість полів парно, щоб переконатись що алгоритм працює коректно. Після того, як результати правильності обрахунку стали успішними, слідувало тестування всіх можливих активних елементів у програмі. Велика кількість натискань на кнопку генерації під ряд спричинила деякі баги, програма видавала помилку, після чого виключалась. У вкладці «Вузли» знайшлась помилка алгоритму генерації вузлів. Всі вузли генерувались у парній кількості. Знайшлась помилка при роботі конкретного поля, при якій це поле завжди вважалось невірним. Всі помилки було виправлено.

Вкладка «IP-адреси». В даному блоці перевірялись: кнопка генерації, робота таймеру, кнопка вибору класової чи безкласової адресації, кнопки вимкнення полів і кнопка оцінки, робота таймеру. Також, важливим було перевірити особливості роботи модуля для контролю знань, таких як:

блокування вкладки після генерації даних, неможливість редагування даних після оцінки. Нижче наведені результати тестування.

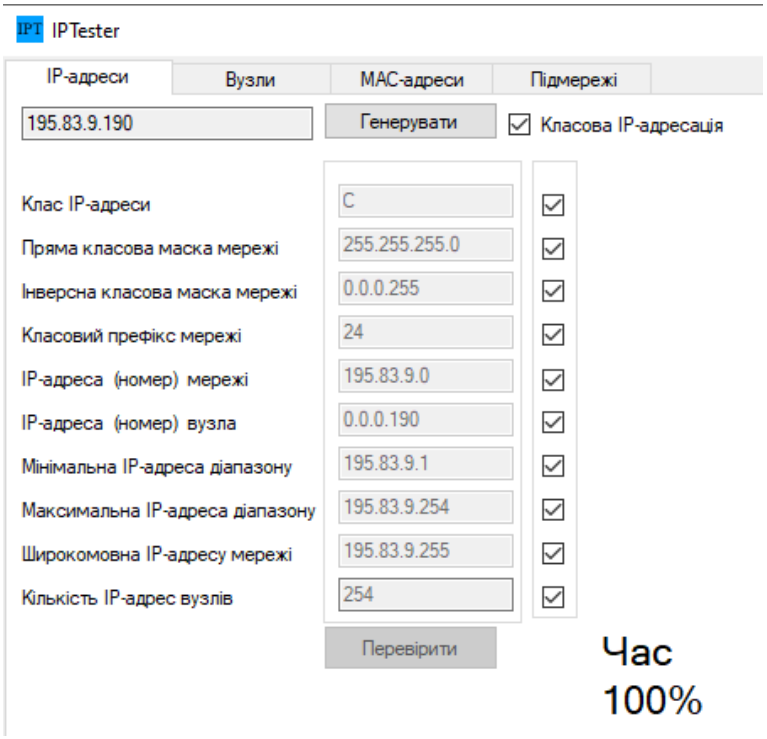


Рис. 3.1. Перевірка коректності обрахунку теми «IP-адреси» класового підходу

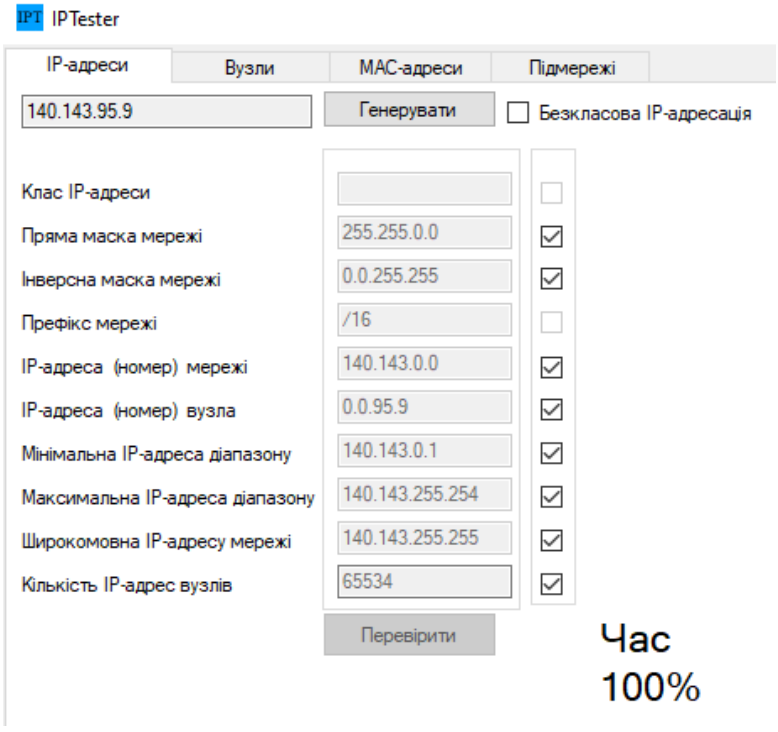


Рис. 3.2. Перевірка коректності обрахунку теми «IP-адреси» безкласового підходу

Вкладка «Вузли». У цій вкладці перевірялись ідентичні елементи, що і у вкладці «IP-адреси». Зокрема важливо було перевірити поля «IP-адреса (номер) мережі» та «IP-адреси (номер) вузла», оскільки алгоритм обрахунку напряму залежить від значень, які будуть введені в ці поля.

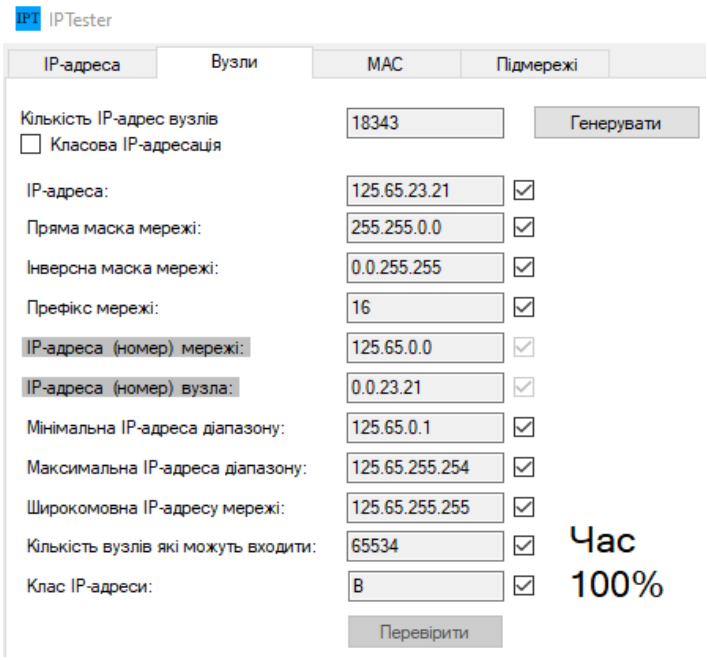


Рис. 3.3. Перевірка коректності обрахунку теми «Вузли» класового підходу

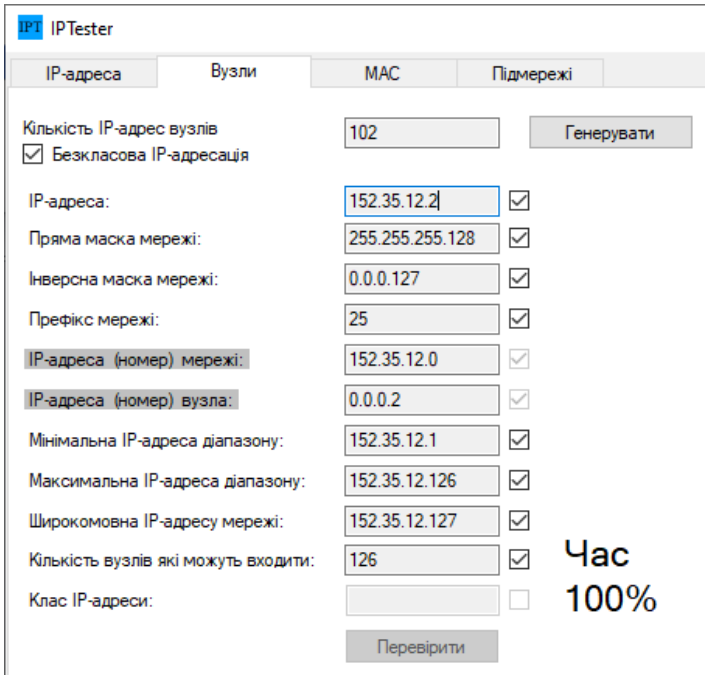


Рис. 3.4. Перевірка коректності обрахунку теми «Вузли» безкласового підходу

Вкладка «MAC-адреси». В цьому блоці набагато менше елементів для перевірки ніж у попередніх вкладках, тож перевірка цієї вкладки пройшла доволі швидко і без помилок.

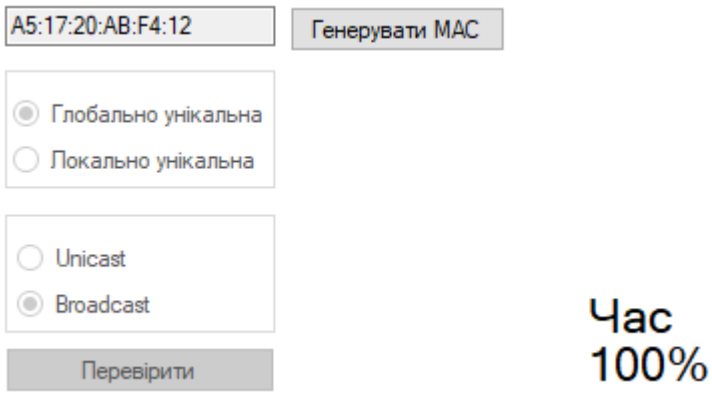


Рис. 3.5. Перевірка коректності обрахунку теми «MAC-адреси»

Вкладка «Підмережі». В цьому блоці 8 завдань. Щоб перевірити кожне завдання окремо, решта завдань вимикались. Важливо було перевірити чи не виникнуть помилки при генерації та обрахунку завдань після додавання нових варіантів у базу даних. Кожне поле в кожному завданні перевірялось окремо. Також перевірялись завдання в парі. Саме в цій вкладці деякі поля оцінювались невірно. Всі помилки, що виникали у вкладці «Підмережі», було виправлено.

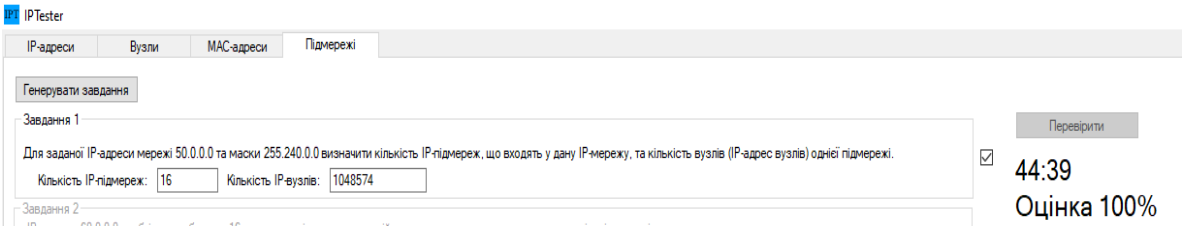


Рис. 3.6. Перевірка коректності обрахунку Завдання 1

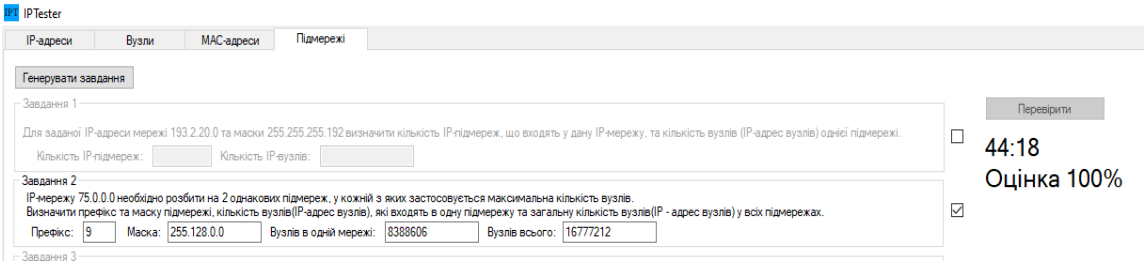


Рис. 3.7. Перевірка коректності обрахунку Завдання 2

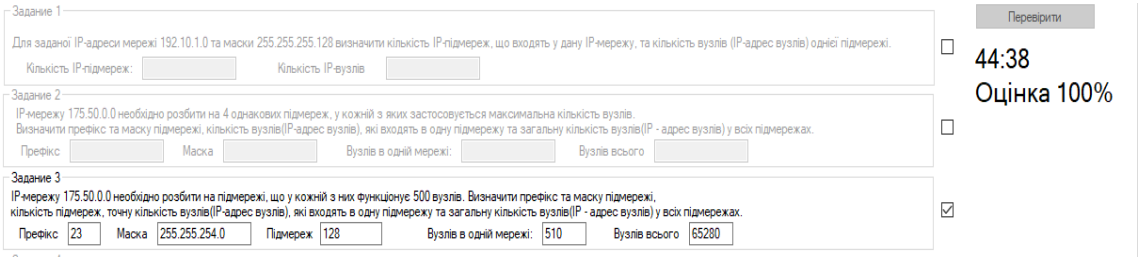


Рис. 3.8. Перевірка коректності обрахунку Завдання 3

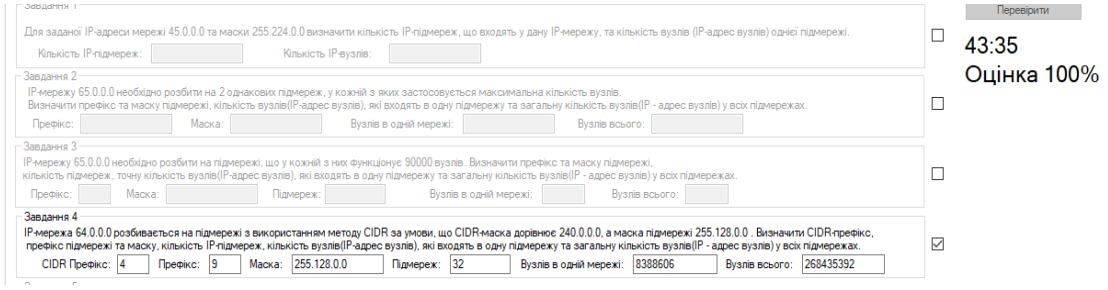


Рис. 3.9. Перевірка коректності обрахунку Завдання 4

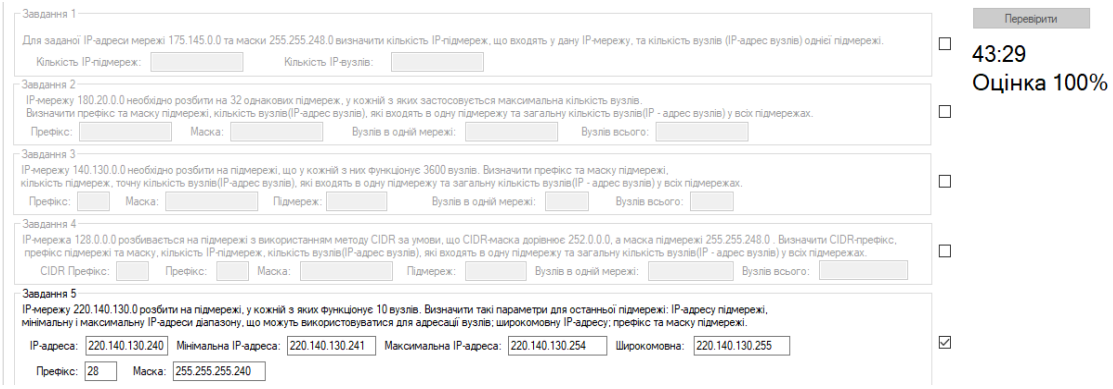


Рис. 3.10. Перевірка коректності обрахунку Завдання 5

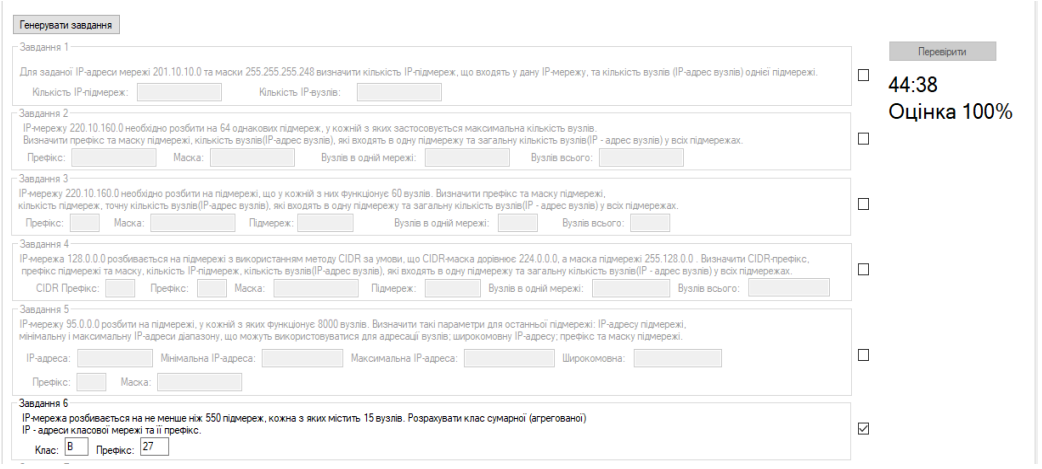


Рис. 3.11. Перевірка коректності обрахунку Завдання 6

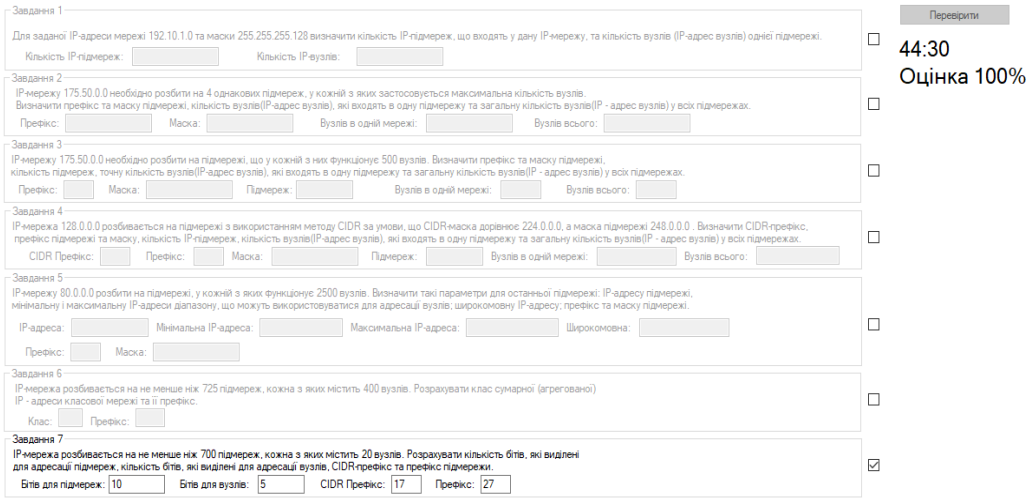


Рис. 3.12. Перевірка коректності обрахунку Завдання 7

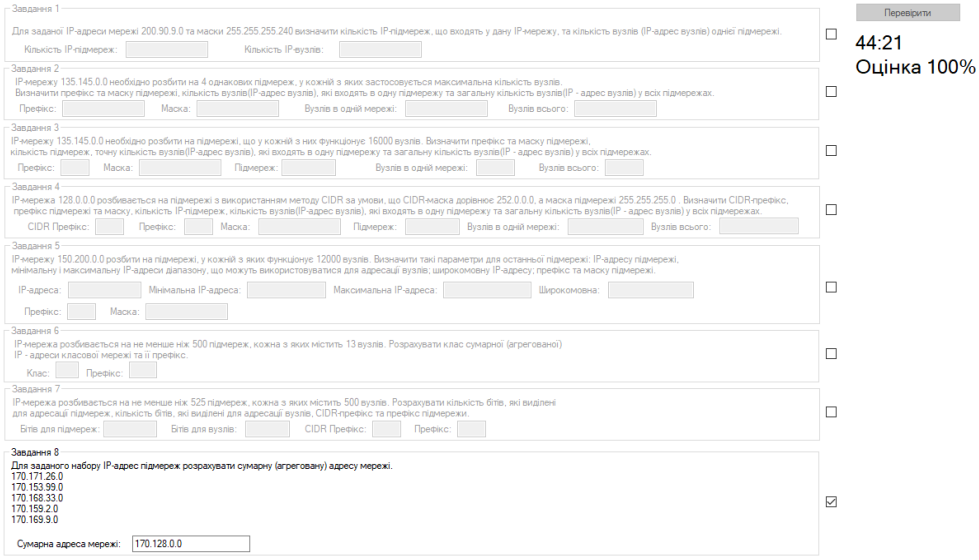


Рис. 3.13. Перевірка коректності обрахунку Завдання 8

На Рис. 3.1-13 показано правильність обрахунку кожного завдання реалізованих в модулю контролю знань. Успішність результатів тестування дозволяє проводити контроль знань реальних користувачів.

3.1.2 Дослідження навчально-тренувальні програми.

Незважаючи на те, що алгоритм обрахунків «IPTeacher» такий самий, як у «IPTester», його все одно потрібно перевірити на виникнення помилок, некоректного підсвічування даних або ж їх генерації. Алгоритм перевірки

навчально-тренувальної програми ідентичний перевірці «IPTester», спираючись на особливості «IPTeacher».

У вкладці «IP-адреси» перевірялись ідентичні елементи що і у одноіменній вкладці у модулі для контролю знань, крім таймеру: кнопка генерації, кнопка вибору класової чи безкласової адресації, кнопки вимкнення полів і кнопка оцінки. Проте вивід оцінки у відсотках змінився на підсвічування правильних і неправильних відповідей. Також, в «IPTeacher» відсутнє блокування вкладок і кількість спроб проходження завдання з тими ж даними необмежена. В навчально-тренувальній програмі появилась можливість редагування полів після оцінки (поля відповіді не блокуються), а отже, важливо було перевірити чи не виникне збій у її роботі після редагування полів відповіді. При перевірці виникали помилки пов'язані із оцінкою. При повторному оцінювані деякі поля вважались неправильними, хоча значення цих полів не підлягали редагуванню. При виправленні цієї помилки виникла нова. При натисканні 2-3 разів на кнопку генерації або оцінки програма припиняла свою роботу повністю. В останній версії програми це було виправлено.

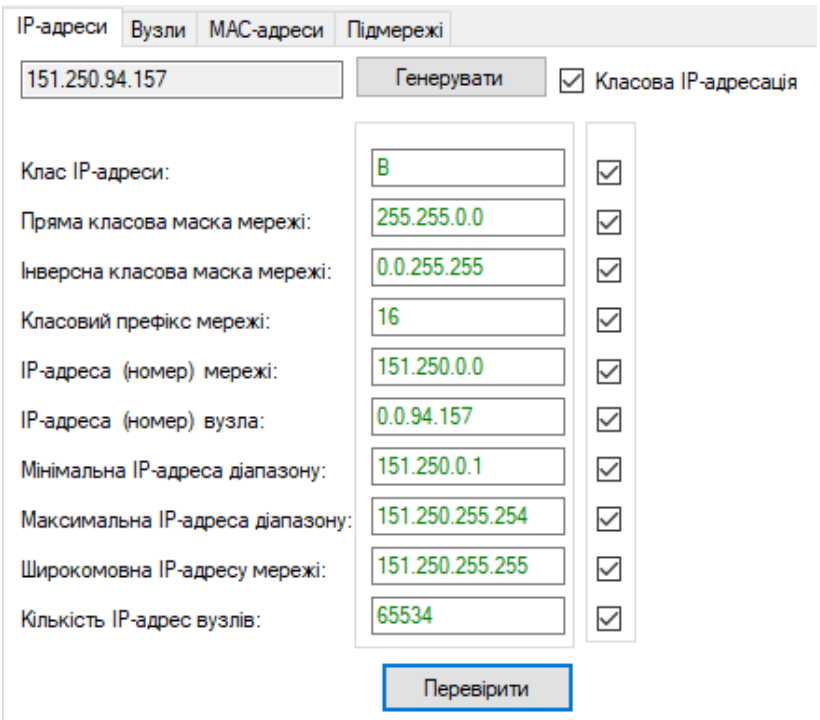


Рис. 3.14. Перевірка коректного обрахунку теми IP-адреси класового підходу

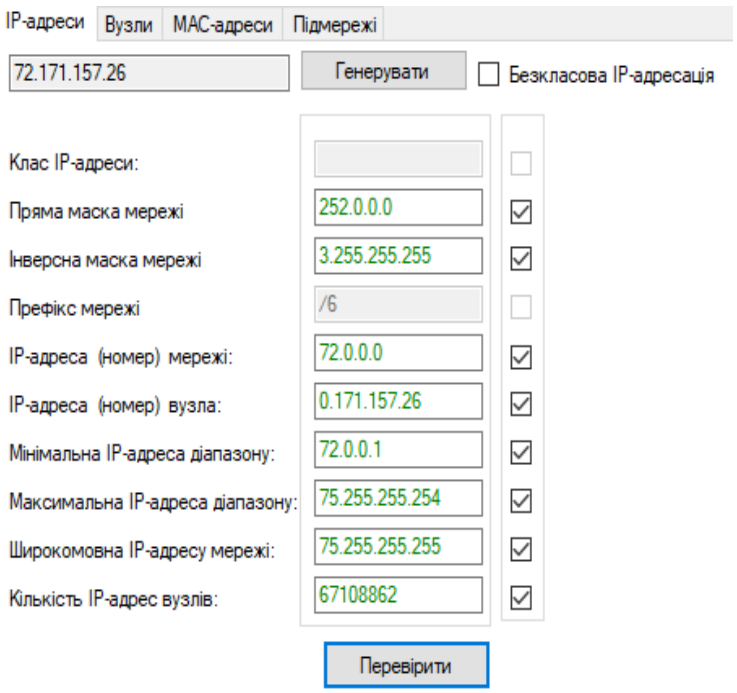


Рис. 3.15. Перевірка коректного обрахунку теми IP-адреси безкласового підходу

Вкладка «Вузли» перевірялась за таким же ж принципом, що і «IP-адреси». В ній практично ідентичні елементи. При перевірці виникали такі ж помилки що і у одноіменній вкладці у «IPTester», які також були успішно виправлені

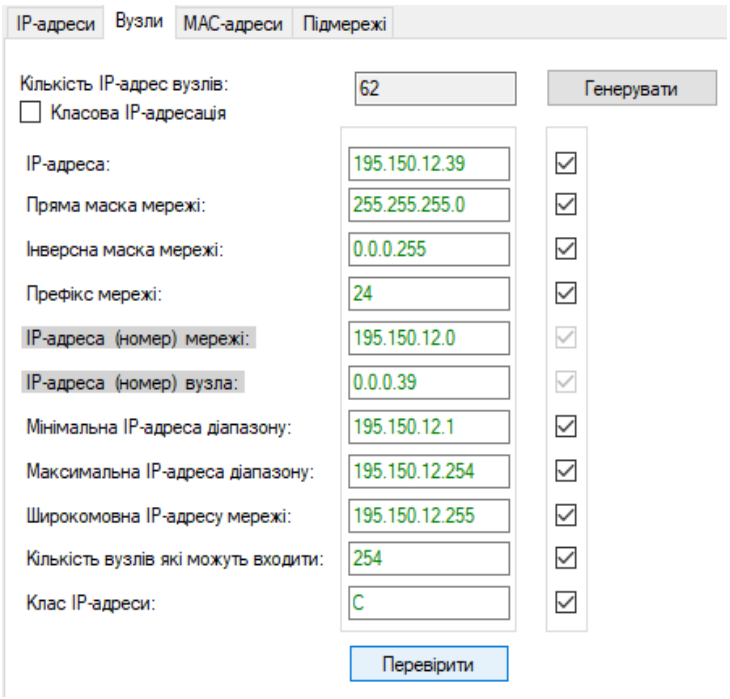


Рис. 3.16. Перевірка коректності обрахунку теми Вузли класового підходу

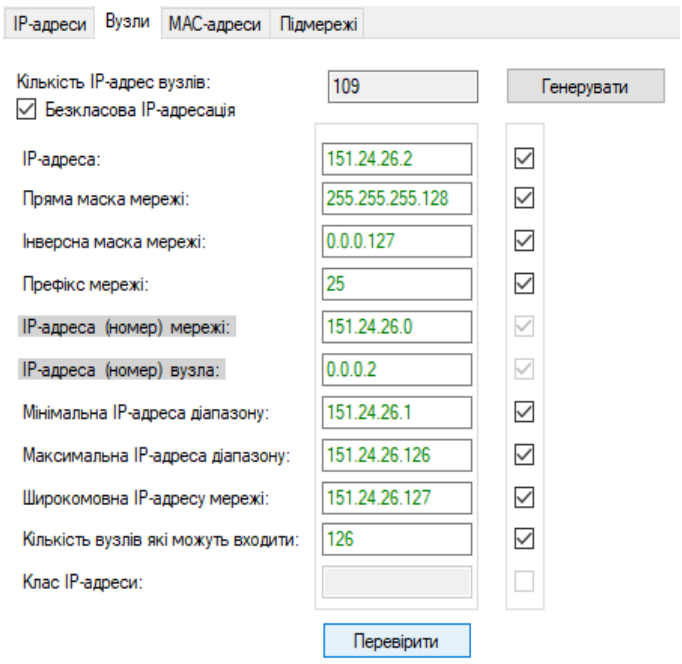


Рис. 3.17. Перевірка коректності обрахунку теми «Вузли» безкласового підходу

Вкладка «MAC-адреси». В цій вкладці набагато менше елементів ніж у решти. Всього лиш 2 поля відповіді, в які, на відміну від решти вкладок, не потрібно вводити значення, а лиш поставити вибір. Перевірка відбулась без помилок.

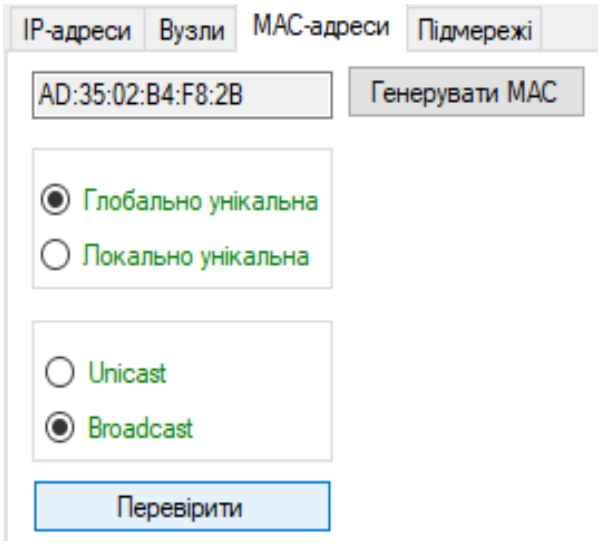


Рис. 3.18. Перевірка коректності обрахунку теми MAC-адреси

Вкладка «Підмережі». Ця вкладка ідеально підійде для випробовування багаторазової оцінки. Це пов’язано із особливістю її роботи. Саме в цій вкладці

відбулось найбільше помилок пов’язаних з оцінкою. Оскільки, програма оцінює всі завдання разом, перевірка відбувалась поступово. Поля заповнювались по черзі, попередні відповіді не видалялись, а після кожного заповнення відбувалась оцінка. Через особливість роботи навчально-тренувальної програми проводити тестування було доволі зручно. Було знайдено помилку в записі варіанту для завдання в базі даних і зразу ж замінено на вірний. Проте, це не виправило помилку оцінювання, коли правильні відповіді спочатку оцінювались коректно, а при повторній перевірці їх колір змінювався на червоний, проте зміни значень не відбувались, але цю помилку також було виправлено.

Завдання 1:

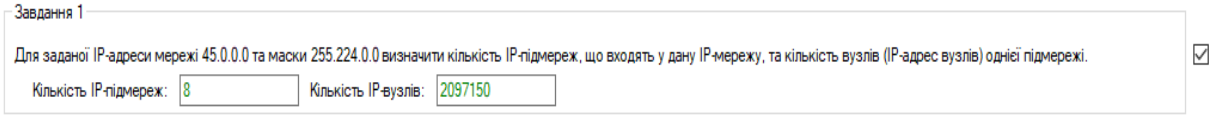


Рис. 3.19. Перевірка коректності обрахунку Завдання 1

Завдання 2:

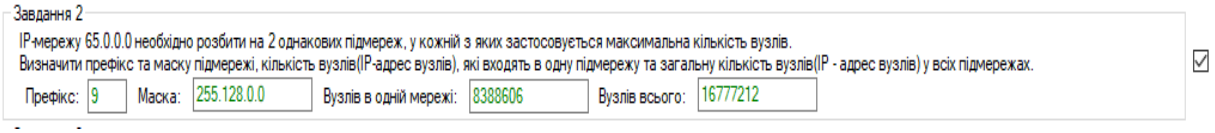


Рис. 3.20. Перевірка коректності обрахунку теми Завдання 2

Завдання 3:

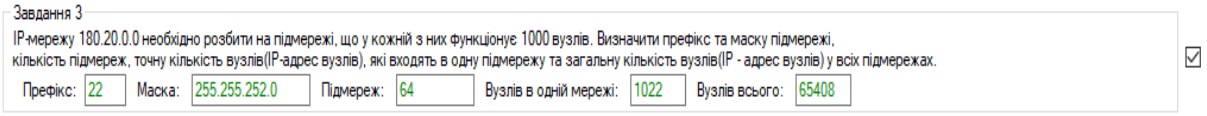


Рис. 3.21. Перевірка правильності обрахунку теми Завдання 3

Завдання 4:

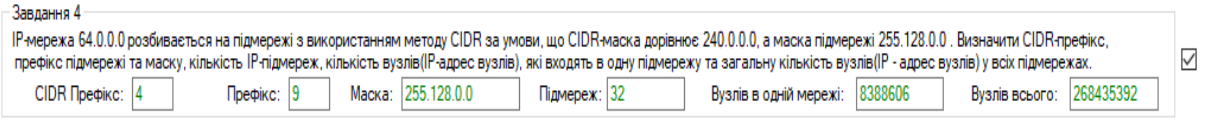


Рис. 3.22. Перевірка правильності обрахунку теми Завдання 4

Завдання 5:

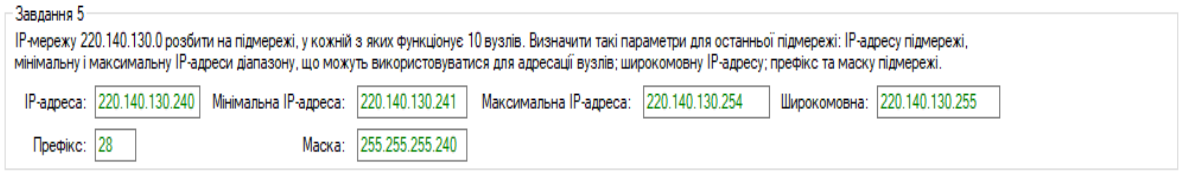


Рис. 3.23. Перевірка правильності обрахунку теми Завдання 5

Завдання 6:

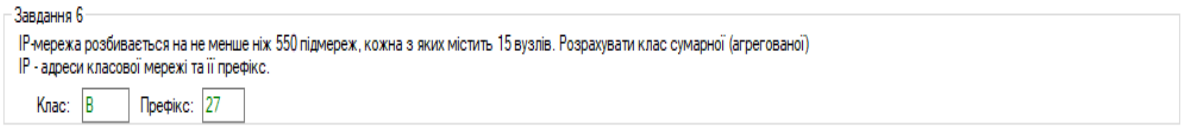


Рис. 3.24. Перевірка правильності обрахунку теми Завдання 6

Завдання 7:

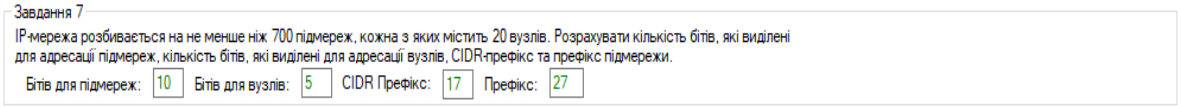


Рис. 3.25. Перевірка правильності обрахунку теми Завдання 7

Завдання 8:

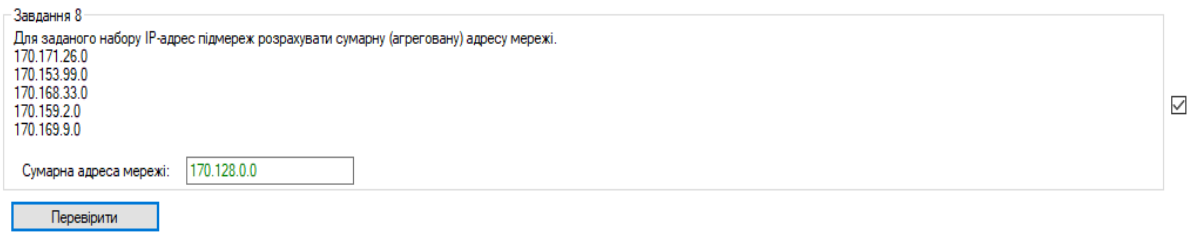


Рис. 3.26. Перевірка правильності обрахунку теми Завдання 8

Перевірка відбулась успішно. Кожен елемент та алгоритми обрахунку працюють коректно. Сама ж інформація, що реалізована для кожної вкладки, доволі легка для засвоєння і чітко пояснює кожне завдання.

3.2 Інструкція користувачу.

3.2.1 Інструкція по користуванню навчально-тренувальною програмою.

Для того ,щоб почати користуватись «IPTeacher», потрібно 2 рази натиснути на запускний файл «IPTeacher». При запуску програми відкриється робоче поле із методичними рекомендаціями першої вкладки. Щоб перейти на іншу вкладку потрібно на неї натиснути. Між вкладками можливо вільно переключатись, якщо ввести відповіді у поля на якійсь з вкладок, то збережуться при поверненні.

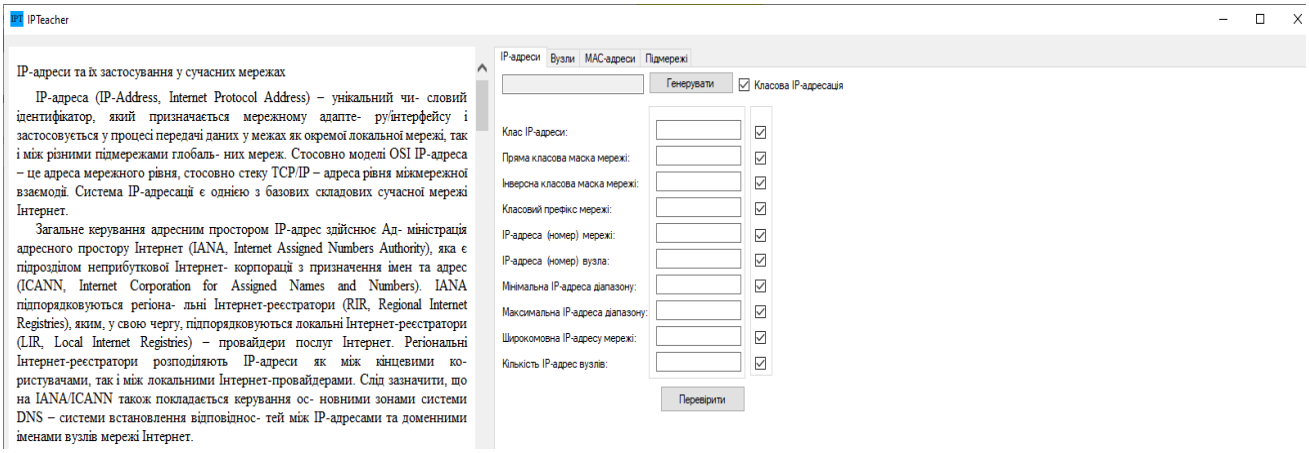


Рис. 3.27. Вигляд робочого поля «IP-адреси»

Перед тим, як почати виконувати завдання у вкладці «IP-адреси» є можливість ознайомитись із теоретичним матеріалом. Він знаходиться в лівій частині програми. Далі слід вибрати класовий або безкласовий підхід (вибрати або зняти вибір у відповідному полі біля кнопки «Генерувати»). По замовчуванню стоїть класовий підхід. Класовий підхід наявний тільки у вкладках «IP-адреси» та «Вузли».

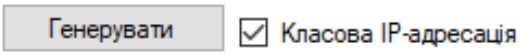


Рис. 3.28. Кнопка переключення класу

При безкласовому підході поле «Клас IP-адреси» вимикається автоматично, а поле «Класовий префікс мережі» змінюється на поле «Префікс мережі» і заповнюється випадковим префіксом.

Для того, щоб згенерувати дані потрібно натиснути кнопку «Генерувати». Згенеровані дані будуть оновлюватись з кожним натисканням на кнопку генерації. Далі можна вводити відповіді у відповідні поля.

IP-адреси, маски, номер вузла, номер мережі тощо, слід вводити через крапку. Кома чи будь-який інший знак буде помилковим. Це стосується кожної вкладки.

Наявна можливість відключити поле відповіді. Для цього потрібно зняти вибір на елементі навпроти поля відповіді, а щоб увімкнути потрібно натиснути на елемент ще раз. Вимкнені поля не будуть оцінюватися.



Рис. 3.29. Вигляд елементів вимкнення полів

Для того, щоб програма перевірила правильність введених відповідей потрібно натиснути кнопку «Перевірити». Перевірка підсвічує правильні відповіді зеленим, а неправильні червоним. Також в «IPTeacher» є можливість відредагувати, яке завгодно поле, якщо воно увімкнуте. Після редагування можна виконати повторну перевірку. В навчально-тренувальній програмі перевірка може виконуватись безліч разів.

Нова генерація даних оновлює попередні відповіді, тож всі написані до того відповіді зникнуть.

У вкладці «Вузли» реалізовані ці ж функції що і у вкладці «IP-адреси», проте є свої особливості.

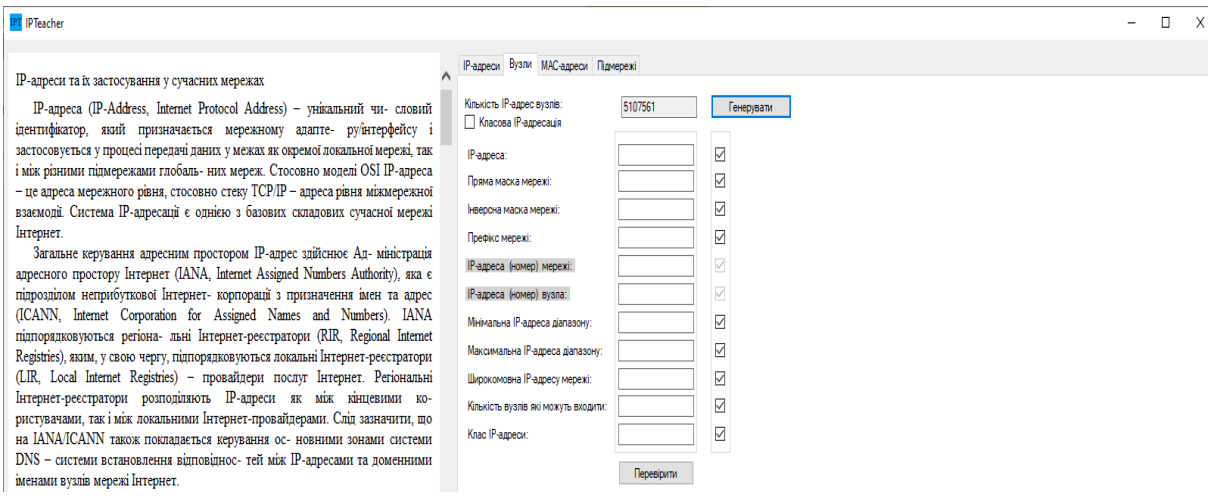


Рис. 3.30. Вигляд робочого поля вкладки «Вузли»

Наприклад, поля «IP-адреса (номер) мережі» та «IP-адреса (номер) вузла» є основними. Тобто, щоб програма успішно перевірила відповіді ці поля повинні бути заповненими, тому що від них залежить обрахунок решти полів. Ці поля виділені і їх вимкнути неможливо. Всі решта елементи ідентичні, як і у вкладці «IP-адреси», з різницею в розташуванні деяких елементів . Елемент переключення класу адресації знаходиться під назвою поля генерації «Кількість IP-адрес вузлів», а поле «Клас IP-адреси» останнім.

У вкладці «MAC-адреси» все доволі просто. В ній реалізовані всього лиш 2 завдання та в кожному завданні 2 варіанти відповіді. Генерація і оцінка відбувається натисканням відповідних кнопок, як і у попередніх вкладках.

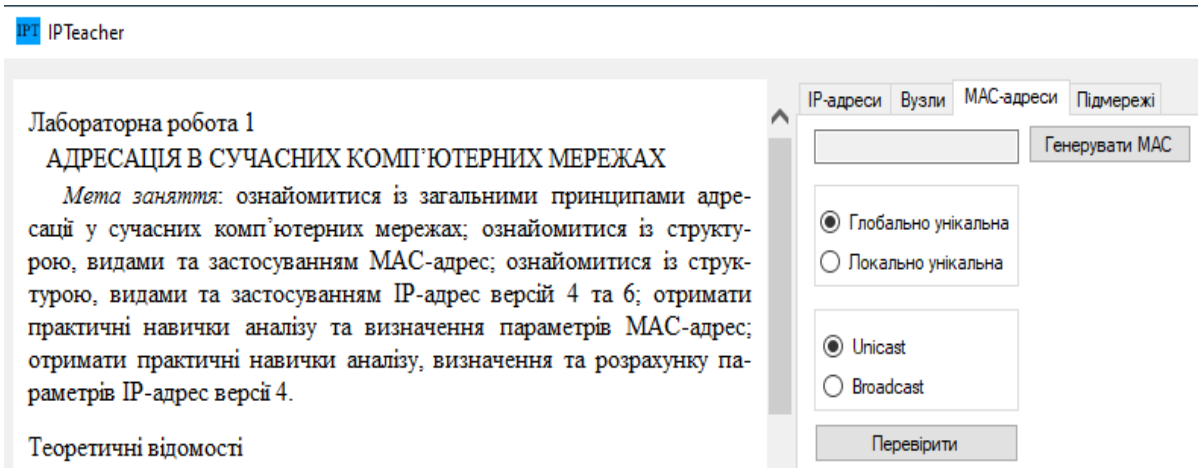


Рис. 3.31. Вигляд робочого поля вкладки «MAC-адреси»

Вкладка «Підмережі» виглядає наступним чином:

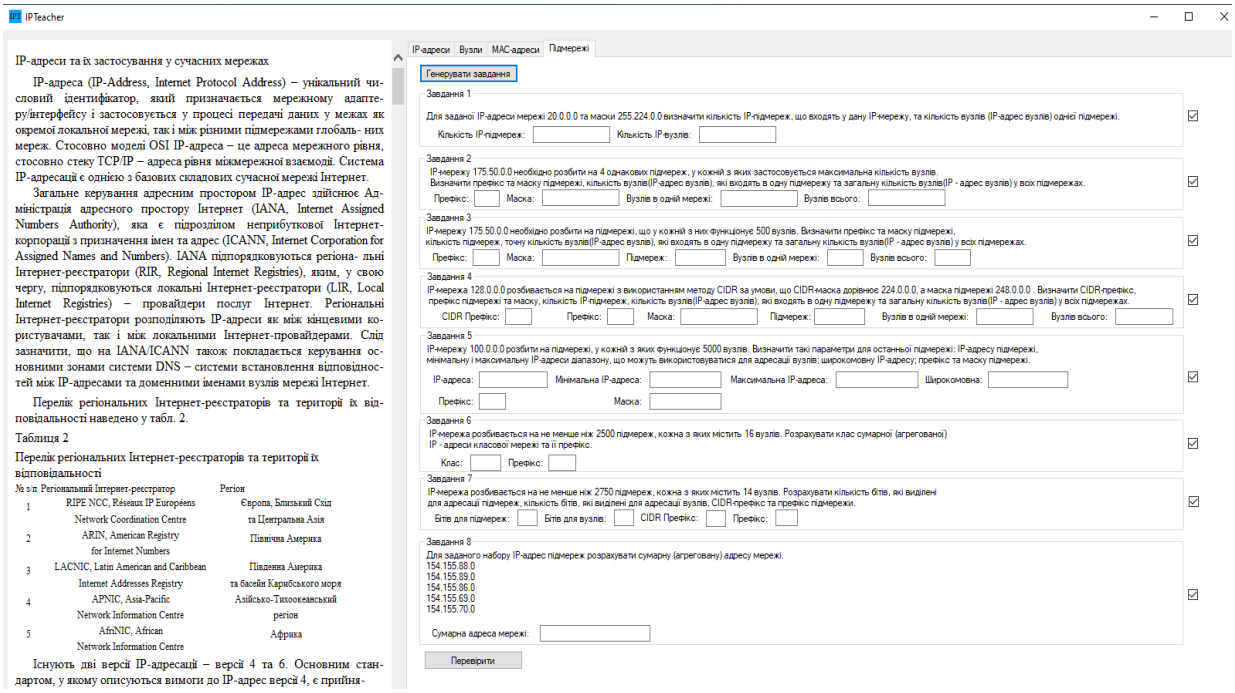


Рис. 3.32. Вигляд робочого поля вкладки «Підмережі»

В даній вкладці реалізований файл для читання, як і в попередніх. На даний момент доступно 8 завдань кожне з яких має певну кількість полів для відповіді.

Генерація, як і в попередніх вкладках відбувається шляхом натискання на кнопку «Генерувати завдання". Кнопка генерації генерує дані для всіх завдань одночасно. Також кнопка генерації скидає попередні відповіді у всіх полях одночасно. У вкладці «Підмережі» реалізовані вимкнення завдань. Елемент вимкнення знаходиться навпроти кожного завдання. Для того, щоб вимкнути завдання, потрібно натиснути на цей елемент. Вимкненні завдання не оцінюються.

В кожному завданні міститься певна кількість полів, які потрібно заповнити. Щоб перевірити їх, потрібно натиснути на кнопку «Перевірити». Оцінка відбувається для всіх завдань одночасно.

3.2.2 Інструкція по користуванню модулем контрою знань.

Для того, щоб почати користуватись «IPTester», потрібно 2 рази натиснути на запускний файл «IPTester». При запуску програми відкриється

робоче поле першої вкладки. Щоб перейти на іншу вкладку потрібно на неї натиснути.

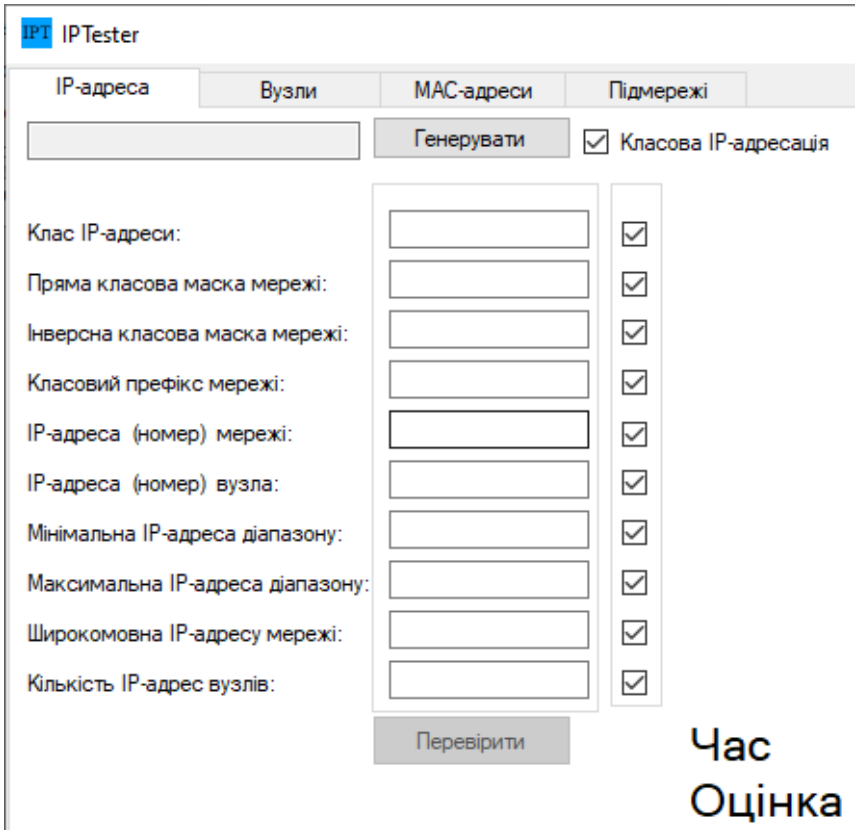


Рис. 3.33. Вигляд робочого поля «IP-адреси»

Далі слід вибрати класовий або безкласовий підхід (вибрати або зняти вибір у відповідному полі біля кнопки «Генерувати»). По замовчуванню стоїть класовий підхід. Класовий підхід наявний тільки у вкладках «IP-адреси» та «Вузли».

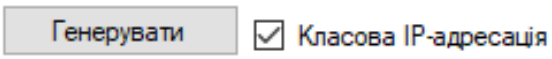


Рис. 3.34. Кнопка переключення класу

При безкласовому підході поле «Клас IP-адреси» вимикається автоматично, а поле «Класовий префікс мережі» змінюється на поле «Префікс мережі» і заповнюється випадковим префіксом.

Для того, щоб згенерувати дані потрібно натиснути кнопку «Генерувати». Згенеровані дані будуть оновлюватись з кожним натисканням на кнопку

генерації. Також, кнопка генерації запускає таймер, за час якого користувачу потрібно впоратись із завданням. Відлік часу відбувається в полі «Час». По завершенню роботи таймера завдання буде оцінено автоматично. При повторному натисканні на кнопку «Генерувати» таймер починає відлік заново. Також, після генерації вкладка блокується і переключення на іншу стає неможливим. Далі можна вводити відповіді у відповідні поля. Введенні відповіді оновляться при натисканні на кнопку «Генерувати». Також, кнопка генерації розблоковує кнопку «Оцінити». За замовчуванням вона заблокована.

Для того, щоб оцінити завдання, потрібно натиснути на кнопку «Оцінити». Після натискання на кнопку оцінки зупиняється таймер і виводиться набраний бал у відсотковому співвідношенні правильних відповідей до кількості полів. Кількість балів виводиться у полі «Оцінка». Поля відповіді блокуються із введеними відповідями після оцінки, що унеможливлює їх редагування. Кожне поле відповіді оцінюється в однакову кількість відсотків. Кнопка «Оцінити» розблоковує вкладку і можна перемикатись між всіма вкладками.

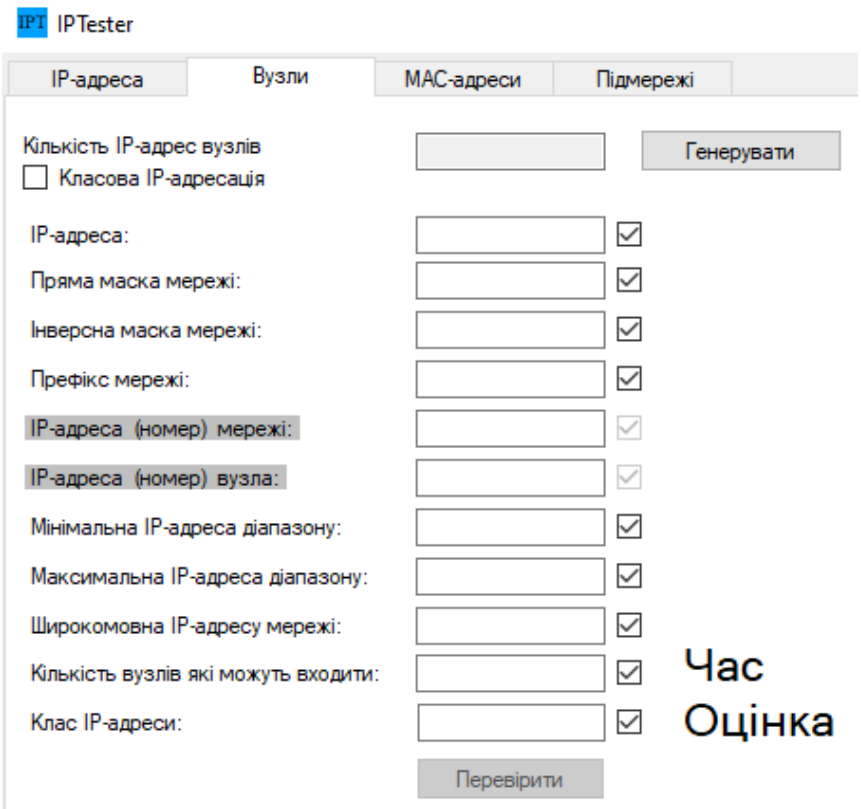


Рис. 3.35. Вигляд робочого поля вкладки «Вузли»

У вкладці «Вузли» реалізовані ці ж функції що і у вкладці «IP-адреси», проте є свої особливості.

Наприклад, поля «IP-адреса (номер) мережі» та «IP-адреса (номер) вузла» є основними. Спочатку потрібно вводити значення в ці поля, тому що від них залежить обрахунок решти полів. Ці поля вимкнути неможливо. Всі решта елементи ідентичні, як і у вкладці «IP-адреси», з різницею в розташуванні елементу переключення класу адресації (він знаходиться під назвою поля генерації «Кількість IP-адрес вузлів»).

У вкладці «MAC-адреси» все доволі просто. В ній реалізовані всього лиш 2 завдання та в кожному завданні 2 варіанти відповіді.

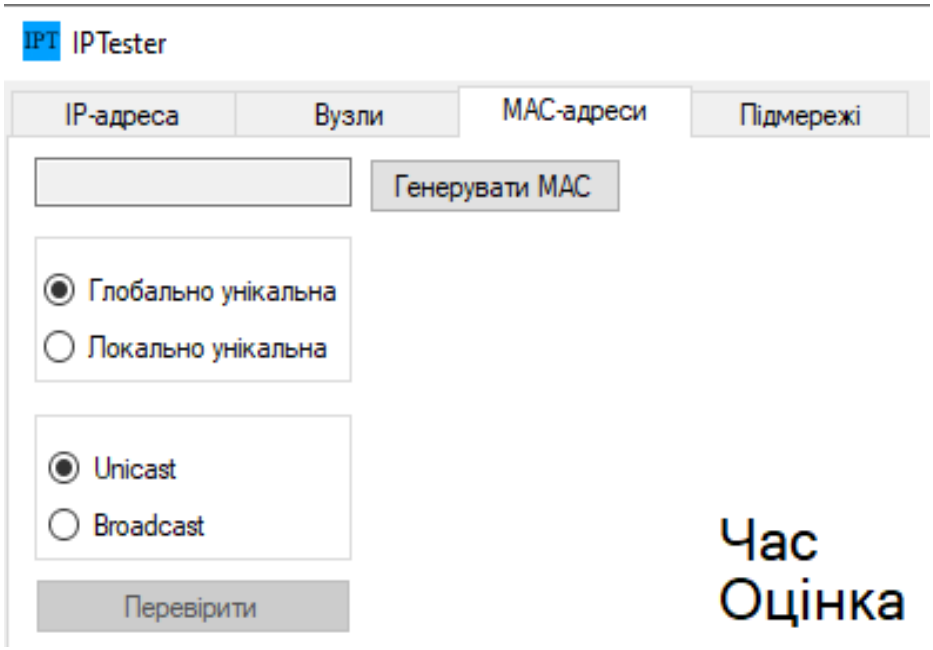


Рис. 3.36. Вигляд робочого поля вкладки «MAC-адреси».

Вкладка «Підмережі». На даний момент доступно 8 завдань, кожне з яких має певну кількість полів для відповіді.

Генерація варіантів, як і в попередніх вкладках, відбувається шляхом натискання на кнопку «Генерувати завдання". Ця кнопка генерує дані для всіх завдань одночасно. Так само, кнопка генерації скидає попередні відповіді у всіх полях одночасно. У вкладці «Підмережі» реалізовані вимкнення завдань. Елемент вимкнення знаходиться навпроти кожного завдання. Для того, щоб

вимкнути завдання потрібно натиснути на цей елемент. Вимкненні завдання не оцінюються.

В кожному завданні міститься певна кількість полів, які потрібно заповнити. Щоб перевірити коректність внесених відповідей, потрібно натиснути на кнопку «Перевірити». Оцінка, як і генерація завдань відбувається для всіх завдань одночасно.

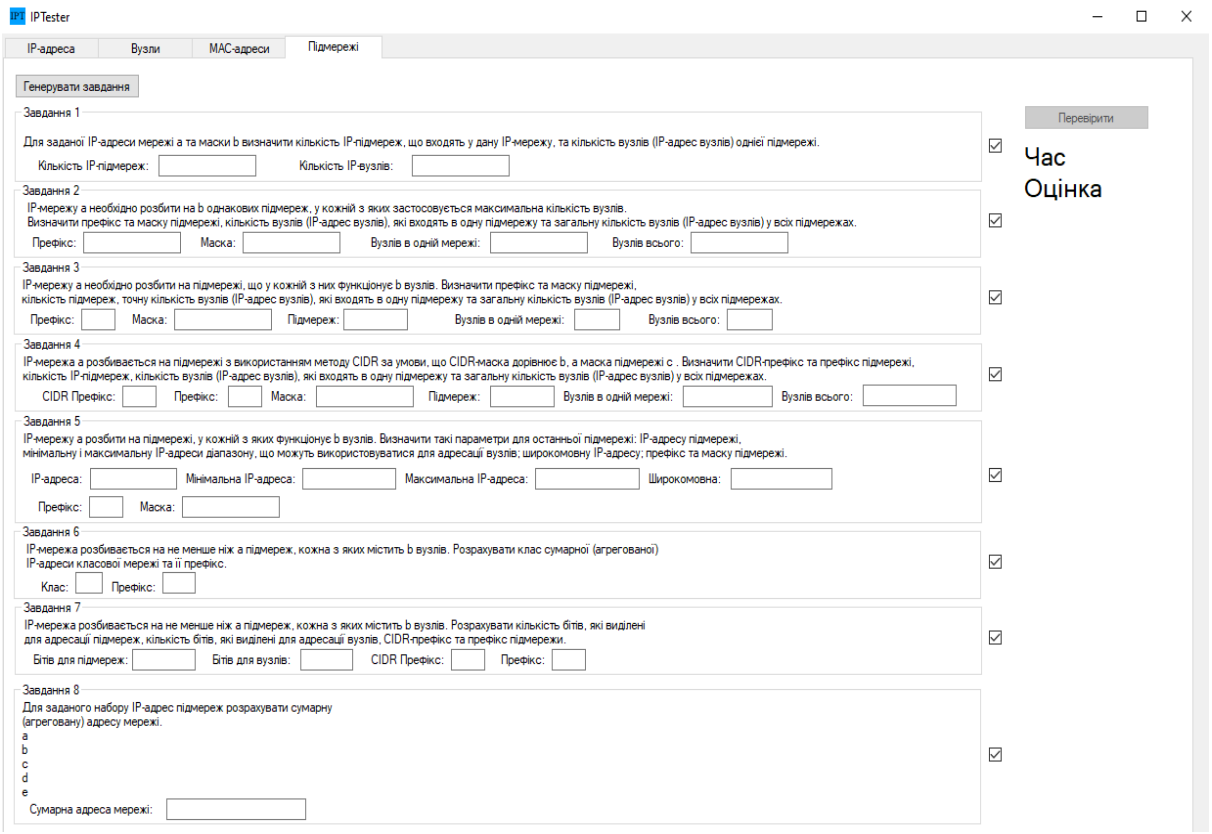


Рис. 3.37. Вигляд робочого поля вкладки «Підмережі»

Висновки по 3 розділу

В третьому розділі було перевірено працездатність програми. Коректність роботи виводу даних, роботи активних елементів та правильність обрахунку відповідей. Перевірена коректність підсвічування відповідей. Було проведено тестування програми на помилки та баги.

Також був проведений детальний інструктаж по користуванню «IPTeacher» та «IPTester». Розказано про особливості роботи кожної з програм, особливості роботи вкладок, кнопок тощо.

ВИСНОВКИ

При виконанні першого розділу кваліфікаційної робити був проведений аналіз навчальних програм в цілому. Проведено аналіз та підтверджено актуальность навчально-тренувальних програм у вигляді комп'ютерних програм та сайтів. Проведено аналіз вимог та особливостей подібних програм. Було розглянуто мову програмування C# та вибрано найоптимальніше середовище розробки для неї. Також було проаналізовано поставлене завдання на розробку, поставлений план виконання проєкту спираючись на попередні пункти.

В другому розділі було розглянуто структуру навчально-тренувальної програми та основних файлів програми. Також, були розглянуті вибрані інструменти для реалізації даної програми. Була розглянута тема розробки програмного забезпечення, необхідні додаткові програми, ресурси та особливості розробки.

В третьому розділі було перевірено працездатність програми. Коректність роботи виводу даних, роботи активних елементів та правильність обрахунку відповідей. Перевірена коректність підсвічування відповідей.

Також був проведений детальний інструктаж по користуванню «IPTeacher» та «IPTester». Розказано про особливості роботи кожної з програм, особливості роботи вкладок, кнопок тощо.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Агуров П. C#. Сборник рецептов. Санкт-Петербург: БХВ, 2007. 432 с.
2. Албахари Д., Албахари Б. C# 3.0. Справочник. Санкт-Петербург: БХВ, 2012. 944 с.
3. Бадд Т. Объектно-ориентированное программирование в действии. Санкт-Петербург: Питер, 1997. 464 с.
4. Бишоп Дж., Хорспул Н. C# в кратком изложении. Москва: Бином. Лаборатория знаний, 2013. 472 с.
5. Бібліотека MSDN розділ – Довідник по C# – [Електронний ресурс] – Режим доступу до ресурсу: https://msdn.microsoft.com/ru-ru/library/618ayhy6.aspx
6. Брауде Э. Технология разработки программного обеспечения. Санкт-Петербург: Питер, 2004. 654 с.
7. Буч Г., Роберт А., Майкл У., Бобби Дж., Коналлен Дж., Келли А. Объектно-ориентированное программирование. – Калифорния: Вильямс, 2008. 514 с.
8. Бьюли А. Изучаем SQL. Санкт-Петербург: Символ-Плюс, 2007. 312 с.
9. Вагнер Б. C# Эффективное программирование. Санкт-Петербург: ЛОРИ, 2013. 320 с.
10. Дейт К. Дж. Введение в системы баз данных = Introduction to Database Systems. — 8-е изд. Калифорния: Вильямс, 2005. 1328 с.
11. Зайченко Ю. Комп'ютерні мережі: Навчальний посібник. Вінниця: Слово, 2003. 286 с.
12. Зиборов В. Visual C# 2012 на примерах. Санкт-Петербург: БХВ, 2013. 480 с.
13. Ишкова Э. C# начала программирования. Москва: БИНОМ. Лаборатория знаний, 2007. 496 с.
14. Лозікова Г.М. Комп'ютерні мережі. Київ: Центр навчальної літератури, 2004. 128 с.
15. Молинаро Э. SQL. Сборник рецептов. Калифорния: O'Reilly, 2009. 672 с.
16. Нейгел К., Ивьен Б., Глинн Дж., Уотсон К., Скиннер М. C# 5.0 и платформа .NET 4.5 для профессионалов = Professional C# 5.0 and .NET 4.5. Москва: «Диалектика», 2013. 1440 с.

17. Павловская Т.А. C#. Программирование на языке высокого уровня: Учебник для вузов. Санкт-Петербург: Питер, 2009. 432 с.
18. Подбельський В. Мова C# Базовий курс. Москва: Інфо Т, 2011. 408 с.
19. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на языке C#. Санкт-Петербург: Питер, 2013. 928 с.
20. Стіллмен Е., Грин Дж. Изучаем C#. 2-е издание = Head First C#, 2ed.Санкт-Петербург: Питер, 2012. 704 с.
21. Троелсен. Е. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е издание = Pro C# 5.0 and the .NET 4.5 Framework, 6th edition. Калифорния: Вильямс, 2013. 1312 с.
22. Фленов М. Библия C#. Санкт-Петербург: БХВ, 2011. 560 с.
23. Хейлсберг А., Торгерсен М., Вилтамут С., Голд П. Язык программирования C#. Классика Computers Science. 4-е издание = C# Programming Language (Covering C# 4.0), 4th Ed. – Санкт-Петербург: Питер., 2012. 784 с.
24. Шилдт Г. C# 4.0: Полное руководство. Калифорния: Вильямс, 2011. 1056 с.
25. C#. Language Specification. Version 3.0 / Copyright. Microsoft Corporation 1999-2007. All Rights Reserved. [Електронний ресурс] – Режим доступу до ресурсу: http://download.microsoft.com/download/3/8/8/388e7205-bc10-4226-b2a8-75351c669b09/CSharp%20Language%20Specification.doc

ДОДАТКИ

Додаток А

Лістинг програм

IPTeacher:

```
namespace IPTester
{
    public partial class Form1
    {
        private void
checkBox1_CheckedChanged_1(object sender,
EventArgs e)
        {
            if (checkBox1.Checked)
            {
                ipClassText.Enabled =
true;
                maxRate += 1;
            }
            else
            {
                {

                    ipPrefix.ForeColor =
Color.Green;
                    Rate++;
                }
                else
                {
                    ipPrefix.ForeColor =
Color.Red;
 = Color.Green;
                    Rate++;
                }
                else
                {
                    hostAddrText.ForeColor
= Color.Red;
                }
            }
            if (checkBox7.Checked &&
minAddrText.Text != "")
            {
                if (minAddrText.Text ==
minAddr)
                {

                    minAddrText.ForeColor
= Color.Green;
                    Rate++;
                }
                else
                {
                    minAddrText.ForeColor
= Color.Red;
                }
            }
            if (checkBox8.Checked &&
maxAddrText.Text != "")
            {
```

Form1.Handler.cs:

```
                if (maxAddrText.Text ==
maxAddr)
                {

                    maxAddrText.ForeColor
= Color.Green;
                    Rate++;
                }
                else
                {
                    maxAddrText.ForeColor
= Color.Red;
                }
            }
            if (checkBox9.Checked &&
broadAddrText.Text != "")
            {
                if (broadAddrText.Text ==
broadcastingAddr)
                {

broadAddrText.ForeColor = Color.Green;
                    Rate++;
                }
                else
                {

broadAddrText.ForeColor = Color.Red;
                }
            }

            if (checkBox10.Checked &&
numOfHostText.Text != "")
            {
                if (numOfHostText.Text ==
numberOfHosts.ToString())
                {

numOfHostText.ForeColor = Color.Green;
                    Rate++;
                }
                else
                {

numOfHostText.ForeColor = Color.Red;
                }
            }
        }
    }
}
```

Form1.Host.cs:

```
        void genHosts()
        {
```

```
            numHosts = 0;
            setStdColorHost();

            clearBox();

            int minPrefix = 0, maxPrefix =
0;
            int nextPrefix = 0;

            if (checkBox11.Checked)
            {
                minPrefix = 0;
                maxPrefix = 30;

                nextPrefix = 1;
            }
            else
            {
                minPrefix = 1;
                maxPrefix = 4;

                nextPrefix = 8;
            }
            prefixHost =
random.Next(minPrefix, maxPrefix);
            if (!checkBox11.Checked)
            {
                prefixHost *= 8;

                if (prefixHost == 8)

textBox12.AccessibleName = "A";
                if (prefixHost == 16)

textBox12.AccessibleName = "B";
                if (prefixHost == 24)

textBox12.AccessibleName = "C";
            }

            uint minValue =
(uint)(Math.Pow(2, 32 - (prefixHost +
nextPrefix))-2);
            if (prefixHost == 24)
            {
                minValue = 1;
            }
            uint maxValue =
(uint)(Math.Pow(2, 32 - prefixHost)-2);

            minValue = (minValue / 2);
            maxValue = (maxValue / 2);

            numHosts =
(uint)random.Next((int)(minValue),
(int)(maxValue))*2;

                    {
                        return false;
                    }
                }
                if (invert)
                {
                    if (addr[i] > 255)
                        return false;
```

Fomr1.Host.Handler.cs:

```
    partial class Form1
    {
        private void
checkBox22_CheckedChanged(object sender,
EventArgs e)
        {
            textBox12.Enabled =
checkBox22.Checked;
            if (textBox12.Enabled)
            {
                hostMaxRate++;
            }
            else
            {
                hostMaxRate--;
            }
        }

        private void
checkBox21_CheckedChanged(object sender,
EventArgs e)
        {
            textBox11.Enabled =
checkBox21.Checked;
            if (textBox11.Enabled)
            {
                hostMaxRate++;
            }
            else
            {
                hostMaxRate--;
            }
        }
        private void
checkBox20_CheckedChanged(object sender,
EventArgs e)
        {
            textBox10.Enabled =
checkBox20.Checked;
            if (textBox10.Enabled)
            {
                hostMaxRate++;
            }
            else
            {
                hostMaxRate--;
            }
        }


        private void
checkBox16_CheckedChanged(object sender,
EventArgs e)
        {
            textBox6.Enabled =
checkBox16.Checked;
            if (textBox6.Enabled)
            {
                hostMaxRate++;
            }
            else
            {
                if(i == 3)
```

```
            }
            else
            {
                hostMaxRate--;
            }
        }
```

Form1.MAC.cs:

```
    partial class Form1
    {
        byte[] currentMAC = new byte[6];
        bool global_local,
unicast_multicast;
        string[] strMACarr = new
string[6];
        string MACstr;


        private void genMAC_Click(object
sender, EventArgs e)
        {
            GenMAC();
        }
        void GenMAC()
        {

            mac1.ForeColor =
System.Drawing.SystemColors.ControlText;
            mac2.ForeColor =
System.Drawing.SystemColors.ControlText;

            for (int i = 0; i < 6; i++)
            {
                currentMAC[i] =
(byte)random.Next(0, 255);
                strMACarr[i] =
Convert.ToString(currentMAC[i], 16);
                if (strMACarr[i].Length ==
1)
                {
                    strMACarr[i] = "0" +
strMACarr[i];
                }
            }
            MACstr =
String.Join(":",strMACarr);
            MACText.Text =
MACstr.ToUpper();
            setFlags();
        }



        private void
sumbit_mac_Click(object sender, EventArgs
e)
        {
            rateMac();
        }
    }
}
```

Task01.cs:

```
        public Task01()
        {
        }

        public void MasktoP()
        {
            int a, b, c, d = 0;
            string a1, b1, c1, d1 = "";
            string str = mask;
            a1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(a1.Length
+ 1);
            b1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(b1.Length
+ 1);
            c1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(c1.Length
+ 1);
            d1 = str;
            a = Convert.ToInt32(a1);
            b = Convert.ToInt32(b1);
            c = Convert.ToInt32(c1);
            d = Convert.ToInt32(d1);
            str = "";

            if (a == 0)
                str += "00000000";
            else
            {
                string tmp =
Convert.ToString(a, 2);
                if (tmp.Length < 8)
                {
                    for (int i = 0; i < 8
- tmp.Length; i++)

            else if (a >= 192 && a <= 223)
                N = 24;
            else
            {
                Console.WriteLine("Цей IP
не відноситься до будь-якого класу.");
                return 0;
            }
            return 1;
        }
        public void final_answer()
        {
            S = P - N;
            H = 32 - P;
            Ks = Math.Pow(2,
Convert.ToDouble(S));
            Kh = Math.Pow(2,
Convert.ToDouble(H))-2;
        }
    }
}
```

Task02.cs:

```
        public void Randoms()
        {
```

```
            Random rnd = new Random();
            DataTable tb = new
DataTable();
            string dbFileName =
"subnet.db3";
            var m_dbConn = new
SQLiteConnection("Data Source=" +
dbFileName + ";Version=3;");
            m_dbConn.Open();
            var sqlQuery = "SELECT * FROM
task02";
            SQLiteDataAdapter adapter =
new SQLiteDataAdapter(sqlQuery, m_dbConn);
            adapter.Fill(tb);
            int tmp = tb.Rows.Count;
            int index = rnd.Next(0, tmp -
1);
            IPaddr =
(string)tb.Rows[index][1];
            Ks =
Convert.ToDouble(tb.Rows[index][2]);
            m_dbConn.Close();
        }


        {
            P = N + S;
            for(int i = 0; i < P; i++)
            {
                maskBin += "1";
            }
            int dopoln = 32 - P;
            for (int i = 0; i < dopoln;
i++)
            {
                maskBin += "0";
            }
            for (int i = 0; i < 4; i ++)
            {
                string tmp =
maskBin.Substring(8*i, 8);
                int byte0 =
Convert.ToInt32(tmp, 2);
                mask +=
Convert.ToString(byte0) + ".";
            }
            mask = mask.Substring(0,
mask.Length-1);
            H = 32 - N - S;
            Kh = Math.Pow(2,
Convert.ToDouble(H)) - 2;
            Khr = Kh * Ks;

        }
    }
}
```

Task03.cs:

```
    public class Task03
    {
        public void Randoms()
        {
            Random rnd = new Random();
            DataTable tb = new
DataTable();
            string dbFileName =
"subnet.db3";
            var m_dbConn = new
SQLiteConnection("Data Source=" +
dbFileName + ";Version=3;");
            m_dbConn.Open();
            var sqlQuery = "SELECT * FROM
task03";
            SQLiteDataAdapter adapter =
new SQLiteDataAdapter(sqlQuery, m_dbConn);
            adapter.Fill(tb);
            int tmp = tb.Rows.Count;
            int index = rnd.Next(0, tmp -
1);
            IPaddr =
(string)tb.Rows[index][1];
            Kh =
Convert.ToDouble(tb.Rows[index][2]);
            m_dbConn.Close();
        }
        public int classIP()
        {
            int a, b, c, d = 0;
            string a1, b1, c1, d1 = "";
            string str = IPaddr;
            a1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(a1.Length
+ 1);
            b1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(b1.Length
+ 1);
            c1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(c1.Length
+ 1);
            d1 = str;

            d = Convert.ToInt32(d1);
            if (a >= 1 && a <= 126)
                N = 8;
            else if (a >= 128 && a <= 191)
                N = 16;
            else if (a >= 192 && a <= 223)
                N = 24;
            else
            {
                Console.
 Convert.ToInt32(tmp, 2);
                mask +=
Convert.ToString(byte0) + ".";
            }
            mask = mask.Substring(0,
mask.Length - 1);
            Ks = Math.Pow(2,
Convert.ToDouble(S));
            Khi = Math.Pow(2,
Convert.ToDouble(H))-2;
            Khr = Ks * Khi;
        }
    }
}
```

Task04.cs:

```
        public void Randoms()
        {
            Random rnd = new Random();
            DataTable tb = new
DataTable();
            string dbFileName =
"subnet.db3";
            var m_dbConn = new
SQLiteConnection("Data Source=" +
dbFileName + ";Version=3;");
            m_dbConn.Open();
            var sqlQuery = "SELECT * FROM
task04";
            SQLiteDataAdapter adapter =
new SQLiteDataAdapter(sqlQuery, m_dbConn);
            adapter.Fill(tb);
            int tmp = tb.Rows.Count;
            int index = rnd.Next(0, tmp -
1);
            IPaddr =
(string)tb.Rows[index][1];
            CIDRmask =
(string)tb.Rows[index][2];
            mask =
(string)tb.Rows[index][3];
            m_dbConn.Close();


        }
        public void MasktoP()
        {
            int a, b, c, d = 0;
            string a1, b1, c1, d1 = "";
            string str = mask;
            a1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(a1.Length
+ 1);
            b1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(b1.Length
+ 1);
            c1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(c1.Length
+ 1);
            d1 = str;
            a = Convert.ToInt32(a1);
            b = Convert.ToInt32(b1);
            c = Convert.ToInt32(c1);
            d = Convert.ToInt32(d1);
            str = "";

            if (a == 0)
                str += "00000000";
            else
            {
                string tmp =
Convert.ToString(a, 2);
                if (tmp.Length < 8)
                {
                    for (int i = 0; i < 8
- tmp.Length; i++)
                        tmp = "0" + tmp;
                }
                str += tmp;
            }
            if (b == 0)
                str += "00000000";
            else
            {
                string tmp =
                string tmp =
Convert.ToString(d, 2);
                if (tmp.Length < 8)
                {
                    for (int i = 0; i < 8
- tmp.Length; i++)
                        tmp = "0" + tmp;
                }
                str += tmp;
        }
        public void final_result(){
            S = P - C;
            H = 32 - P;
            Kp = Math.Pow(2,
Convert.ToDouble(S));
            Kv = Math.Pow(2,
Convert.ToDouble(H)) - 2;
            Kz = Kp * Kv;
        }
    }
}
```

Task05.cs:

```
        public void Randoms()
        {
            Random rnd = new Random();
            DataTable tb = new
DataTable();
            string dbFileName =
"subnet.db3";
            var m_dbConn = new
SQLiteConnection("Data Source=" +
dbFileName + ";Version=3;");
            m_dbConn.Open();
            var sqlQuery = "SELECT * FROM
task05";
            SQLiteDataAdapter adapter =
new SQLiteDataAdapter(sqlQuery, m_dbConn);
            adapter.Fill(tb);
            int tmp = tb.Rows.Count;
            int index = rnd.Next(0, tmp -
1);
            IPaddr =
(string)tb.Rows[index][1];
            Kv =
Convert.ToDouble(tb.Rows[index][2]);
            m_dbConn.Close();
        }
            string str = IP;
            a1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(a1.Length
+ 1);
            b1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(b1.Length
+ 1);
```

```
            c1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(c1.Length
+ 1);
            d1 = str;
            a = Convert.ToInt32(a1);
            b = Convert.ToInt32(b1);
            c = Convert.ToInt32(c1);
            d = Convert.ToInt32(d1);
            str = "";

            if (a == 0)
                str += "00000000";
            else
            {
                string tmp =
Convert.ToString(a, 2);
                if (tmp.Length < 8)
                {
                    for (int i = 0; i < 8
- tmp.Length; i++)
                        tmp = "0" + tmp;

                }
                str += tmp;
            }
            if (b == 0)
                str += "00000000";
            else
            {
                string tmp =
Convert.ToString(b, 2);
                if (tmp.Length < 8)
                {
                    for (int i = 0; i < 8
- tmp.Length; i++)
                        tmp = "0" + tmp;
                }
                str += tmp;
            }
            if (c == 0)
                str += "00000000";
            else
            {
                string tmp =
Convert.ToString(c, 2);
                if (tmp.Length < 8)
                {
                    for (int i = 0; i < 8
- tmp.Length; i++)
                        tmp = "0" + tmp;
                }
                str += tmp;
            }
            if (d == 0)
                str += "00000000";
            else
            {
                string tmp =
Convert.ToString(d, 2);
                if (tmp.Length < 8)
                {
                    for (int i = 0; i < 8
- tmp.Length; i++)
                        tmp = "0" + tmp;
                }
                str += tmp;
            }
            return str;
        }
```

Task06.cs:

```
        public void Randoms()
        {
            Random rnd = new Random();
            DataTable tb = new
DataTable();
            string dbFileName =
"subnet.db3";
            var m_dbConn = new
SQLiteConnection("Data Source=" +
dbFileName + ";Version=3;");
            m_dbConn.Open();
            var sqlQuery = "SELECT * FROM
task06";
            SQLiteDataAdapter adapter =
new SQLiteDataAdapter(sqlQuery, m_dbConn);
            adapter.Fill(tb);
            int tmp = tb.Rows.Count;
            int index = rnd.Next(0, tmp -
1);
            Kv =
Convert.ToDouble(tb.Rows[index][1]);
            Kp =
Convert.ToDouble(tb.Rows[index][2]);
            m_dbConn.Close();
        }
            public void final_result()
        {
            int Y = Convert.ToInt32(Kp) -
1;
            string Y2 =
Convert.ToString(Y, 2);
            S = Y2.Length;
            int X = Convert.ToInt32(Kv) +
2 - 1;
            string X2 =
Convert.ToString(X, 2);
            H = X2.Length;
            double Kp1, Kp2, Kp3 = 0;
            double Kdif = 0;
            int Stmp1 = -1;
            int Stmp2 = -1;
            int Stmp3 = -1;
            int N = 0;
            if ((8 + S + H)<=32)
            {
                Stmp1 = (32 - 8 - H);
                Kp1 = Math.Pow(2, Stmp1);
                Kdif = Kp1-Kp;
                N = 8;
            }
            if ((16 + S + H) <= 32)
            {
                Stmp2 = (32 - 16 - H);
                Kp2 = Math.Pow(2, Stmp2);
                if ((Kp2 - Kp) < Kdif)
                {
                    N = 16;
```

```
                Kdif = Kp2 - Kp;
            }
        }
        if ((24 + S + H) <= 32)
        {
            Stmp3 = (32 - 24 - H);
            Kp3 = Math.Pow(2, Stmp3);
            if ((Kp3 - Kp) < Kdif)
            {
                Kdif = Kp3 - Kp;
                N = 24;
            }
        }
        double Soptimal =
Math.Log(Kdif+Kp, 2);
        S = (int)Soptimal;
        P = N + S;
        if (N == 8)
            classv = "A";
        if (N == 16)
            classv = "B";
        if (N == 24)
            classv = "C";

    }
  }
}
```

Task07.cs:

```
    public void Randoms()
    {
        Random rnd = new Random();
        DataTable tb = new
DataTable();
        string dbFileName =
"subnet.db3";
        var m_dbConn = new
SQLiteConnection("Data Source=" +
dbFileName + ";Version=3;");
        m_dbConn.Open();
        var sqlQuery = "SELECT * FROM
task07";
        SQLiteDataAdapter adapter =
new SQLiteDataAdapter(sqlQuery, m_dbConn);
        adapter.Fill(tb);
        int tmp = tb.Rows.Count;
        int index = rnd.Next(0, tmp -
1);
        Kv =
Convert.ToDouble(tb.Rows[index][1]);
        Kp =
Convert.ToDouble(tb.Rows[index][2]);
        m_dbConn.Close();
    }
        public void final_result()
    {
        int Y = Convert.ToInt32(Kp) -
1;
        string Y2 =
Convert.ToString(Y, 2);
        S = Y2.Length;
        int X = Convert.ToInt32(Kv) +
2 - 1;
        string X2 =
Convert.ToString(X, 2);
        H = X2.Length;
        C = 32 - H - S;
        P = C + S;
    }
  }
}
```

Task08.cs:

```
    class Task08
    {
        public string IPaddr1 = "";
        public string IPaddr2 = "";
        public string IPaddr3 = "";
        public string IPaddr4 = "";
        public string IPaddr5 = "";
        public int P = 0;
        public string result = "";



        public void Randoms()
        {
            Random rnd = new Random();
            DataTable tb = new
DataTable();
            string dbFileName =
"subnet.db3";
            var m_dbConn = new
SQLiteConnection("Data Source=" +
dbFileName + ";Version=3;");
            m_dbConn.Open();
            var sqlQuery = "SELECT * FROM
task08";
            SQLiteDataAdapter adapter =
new SQLiteDataAdapter(sqlQuery, m_dbConn);
            adapter.Fill(tb);
            int tmp = tb.Rows.Count;
            int index = rnd.Next(0, tmp -
1);
            IPaddr1 =
(string)tb.Rows[index][1];
            IPaddr2 =
(string)tb.Rows[index][2];
            IPaddr3 =
(string)tb.Rows[index][3];
            IPaddr4 =
(string)tb.Rows[index][4];
            IPaddr5 =
(string)tb.Rows[index][5];
            m_dbConn.Close();


        }
            string IPaddrToBin(string IP)
        {
            int a, b, c, d = 0;
            string a1, b1, c1, d1 = "";
            string str = IP;
            a1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(a1.Length
+ 1);
            b1 = str.Substring(0,
str.IndexOf("."));
```

```
            str = str.Substring(b1.Length
+ 1);
            c1 = str.Substring(0,
str.IndexOf("."));
            str = str.Substring(c1.Length
+ 1);
            d1 = str;
            a = Convert.ToInt32(a1);
            b = Convert.ToInt32(b1);
            c = Convert.ToInt32(c1);
            d = Convert.ToInt32(d1);
            str = "";

            if (a == 0)
                str += "00000000";
            else
            {
                string tmp =
Convert.ToString(a, 2);
                if (tmp.Length < 8)
                {
                    for (int i = 0; i < 8
- tmp.Length; i++)
                        tmp = "0" + tmp;

                }
                str += tmp;
            }
            if (b == 0)
                str += "00000000";
            else
            {
                string tmp =
Convert.ToString(b, 2);
                if (tmp.Length < 8)
                {
                    for (int i = 0; i < 8
- tmp.Length; i++)
                        tmp = "0" + tmp;
                }
                str += tmp;
            }
            if (c == 0)
                str += "00000000";
            else
            {
                string tmp =
Convert.ToString(c, 2);
                if (tmp.Length < 8)
                {
                    for (int i = 0; i < 8
- tmp.Length; i++)
                        tmp = "0" + tmp;
                }
                str += tmp;
            }
            if (d == 0)
                str += "00000000";
            else
            {
                string tmp =
Convert.ToString(d, 2);
                if (tmp.Length < 8)
                {
                    for (int i = 0; i < 8
- tmp.Length; i++)
                        tmp = "0" + tmp;
                }
                str += tmp;
                }
            }
```

IPTESTER:

Form1.Handler.cs:

```
        private void
inverseMaskText_TextChanged(object sender,
EventArgs e)
        {
            if (inverseMaskText.Text ==
ipInverseMask)
            {
                inverseMaskText.ForeColor
= Color.Green;
            }
            else
            {
                inverseMaskText.ForeColor
= System.Drawing.SystemColors.WindowText;
            }
        }

        private void
hostAddrText_TextChanged(object sender,
EventArgs e)
        {
            if (hostAddrText.Text ==
hostAddr)
            {
            if (broadAddrText.Text ==
broadcastingAddr)
            sumbit.Enabled = false;
            Rate = 0;
            if (
                else
                {

//netmaskText.ForeColor = Color.Red;
                }
            }
            if (checkBox3.Checked)
            {
                if (inverseMaskText.Text
== ipInverseMask)
                {

//ipClassText.ForeColor =
System.Drawing.SystemColors.WindowText;
                    Rate++;
                }
                else
                {

            if (checkBox5.Checked)
            {
                if (netAddrText.Text ==
netAddr)
                {
```

```
//ipClassText.ForeColor =
System.Drawing.SystemColors.WindowText;
                    Rate++;
                }
                else
                {

//netAddrText.ForeColor = Color.Red;

}
```

Form1.Host.cs:

```
    partial class Form1
    {

        void genHosts()
        {
            numHosts = 0;
            unblockBox();
            lockedTab = true;
            timerStart();
            clearBox();

            int minPrefix = 0, maxPrefix =
0;
            int nextPrefix = 0;

            if (checkBox11.Checked)
            {
                minPrefix = 0;
                maxPrefix = 30;

                nextPrefix = 1;
            }
            else
            {
                minPrefix = 1;
                maxPrefix = 4;

                nextPrefix = 8;
            }
            prefixHost =
random.Next(minPrefix, maxPrefix);
            if (!checkBox11.Checked)
            {
                prefixHost *= 8;

                if (prefixHost == 8)

textBox12.AccessibleName = "A";
                if (prefixHost == 16)

textBox12.AccessibleName = "B";
                if (prefixHost == 24)

textBox12.AccessibleName = "C";
            }

            uint minValue =
(uint)(Math.Pow(2, 32 - (prefixHost +
nextPrefix)) - 2);
            if (prefixHost == 24)
            {
                minValue = 1;
        bool validNetAddr(int[] addr,
short[] mask, bool invert)
        {
            for (int i = 0; i <
mask.Length; i++)
            {
                if (invert)
                {
                    if (((~mask[i]) &
addr[i]) != 0)
                    {
                        return false;
                    }
                }
                else
                {
                    if (((mask[i]) &
addr[i]) != 0)
                    {
                        return false;
                    }
                }
            }
            return true;
        }
        private void
genHostBtn_Click(object sender, EventArgs
e)
        {
            genHosts();
        }


    }
}
```

Form1.Host.Handler.cs:

```
    partial class Form1
    {


            genHosts();
            if (!checkBox11.Checked)
            {
                checkBox11.Text = "Класова
IP-адресація";
                checkBox22.Checked = true;
                checkBox22.Enabled = true;
            }
            else
            {
                checkBox11.Text =
"Безкласова IP-адресація";
                checkBox22.Checked =
false;
                checkBox22.Enabled =
false;
            }
        }

        bool validationBox(ref TextBox
box)
        {
```

```
            if (box.Text ==
box.AccessibleName)
                return true;
            return false;
        }

        private void
textBox2_TextChanged(object sender,
EventArgs e)
        {
            setCorrectMask();
        }

        private void
textBox3_TextChanged(object sender,
EventArgs e)
        {
            setCorrectMask();
        }


        private void
textBox5_TextChanged(object sender,
EventArgs e)
        {
            validatorNetAddr();
            setIpAddrValid();

            setMinMaxBroadAddr();
        }

        private void
textBox6_TextChanged(object sender,
EventArgs e)
        {
            validatorHostAddr();
            setIpAddrValid();

            setMinMaxBroadAddr();

        }
        private void
sumbit_host_Click(object sender, EventArgs
e)
        {
            rateHostMeth();
        }
    }
}
```

Form1.MAC.cs:

```
IPTester
{
    partial class Form1
    {
        void GenMAC()
        {
            mac1.Enabled = true;
            mac2.Enabled = true;
            lockedTab = true;
            timerStart();

            for (int i = 0; i < 6; i++)
            {
                currentMAC[i] =
(byte)random.Next(0, 255);
                strMACarr[i] =
Convert.ToString(currentMAC[i], 16);
                if (strMACarr[i].Length ==
1)
                {
                    strMACarr[i] = "0" +
strMACarr[i];
                }
            }
            MACstr =
String.Join(":",strMACarr);
            MACText.Text =
MACstr.ToUpper();
            setFlags();

            sumbit_mac.Enabled = true;
        }
        private void rateMac()
        {
            mac1.Enabled = false;
            mac2.Enabled = false;
            lockedTab = false;
            timerStop();
            sumbit_mac.Enabled = false;
            float rate = 0;
            if (globalUnic.Checked ==
global_local)
                rate += 1;
            if (unicast.Checked ==
unicast_multicast)
                rate += 1;
            rateMAC.Text = (int)((rate /
2) * 100) + "%";
        }

        private void
sumbit_mac_Click(object sender, EventArgs
e)
        {
            rateMac();
        }
    }
}
```