МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ
(підпис)
09 червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерних наук,
освітньо- професійної програми «Інформатика»
на тему: «Графічний інтерфейс налаштування IPv6 адресації у локальній мережі Ethernet»
здобувача групи ІН-93 Карпенка Олександр Володимировича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Олександр КАРПЕНКО
(підпис)

Керівник
старший викладач,
кандидат фізико-математичних наук Дмитро ВЕЛИКОДНИЙ _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-93 Карпенка Олександра Володимировича

1. Тема роботи: «Графічний інтерфейс налаштування IPv6 адресації у локальній мережі Ethernet»
затверджую наказом по СумДУ від *«01» червня 2023 р. № 0475-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз предметної області, аналіз цільової аудиторії, формування та визначення мети дослідження. 2) Розгляд технологій, що використовуються для створення додатків, аналіз інструменту для створення схем. 3) Розроблення додатку, що генерує команди для роутерів. 4) Аналіз отриманих результатів та можливостей покращення додатку.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз предметної області, аналіз цільової аудиторії, формування та визначення мети дослідження.*		
2	*Розгляд технологій, що використовуються для створення додатків, аналіз інструменту для створення схем.*		
3	*Розроблення додатку, що генерує команди для роутерів.*		
4	*Аналіз отриманих результатів та можливостей покращення додатку.*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 49 стр., 25 рис., 1 табл., 3 додатків, 17 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки вона зосереджена на розробці програм для впровадження IPv6 адрес в локальній мережі.

Об'єкт дослідження – процес впровадження IPv6 адрес в локальну мережу.

Предмет дослідження – методи розробки графічного інтерфейсу для налаштування локальних мереж IPv6.

Мета роботи – розробка програми що генерує команди для роутерів.

Методи дослідження – моделі налаштування локальних мереж IPv6 та алгоритми побудови програмного застосунку.

Результати – розроблено графічний інтерфейс для налаштування локальних мереж IPv6, який дозволяє генерувати файл з командами для налаштування роутерів в локальній мережі з можливістю генерації IPv6 адрес.

ПРОГРАМНИЙ ЗАСТОСУНОК, ЛОКАЛЬНА МЕРЕЖА IPV6, C#, WINDOWS FORMS, НАЛАШТУВАННЯ РОУТЕРІВ, ГЕНЕРАЦІЯ АДРЕС

ЗМІСТ

ВСТУП

Актуальність. Тема кваліфікаційної роботи є актуальною, оскільки визначає зростаючий попит на індивідуальні адреси. Використання програми для налаштування роутерів в локальній мережі має потенціал поліпшити доступність, ефективність та зручність для користувачів, провайдерів, що займаються налаштуванням локальних мереж.

Об’єкт дослідження. Процес впровадження IPv6 адрес в локальну мережу.

Предмет дослідження. Методи розробки графічного інтерфейсу для налаштування локальних мереж IPv6.

Гіпотеза. Полегшення налаштування локальних мереж IPv6 можна досягнути шляхом використання інформаційної технології, що реалізує програмний застосунок для створення команд для роутерів.

Наукова новизна. Описане у даній роботі програмне рішення дозволить досягти більшої зручності в налаштуванні локальних мереж IPv6, дозволяючи автоматизувати внутрішні процеси компаній.

Структура. Дана робота складається зі вступу, аналітичного огляду, вибіру методу розв’язання поставленої задачі, опису програмної реалізації, висновків, списку використаних джерел та додатків.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Сучасний стан

У рамках кваліфікаційної роботи бакалавра розглядається тема розробки графічного інтерфейсу для налаштування IPv6 адресації у локальній мережі Ethernet. Перед тим, як увійти у деталі самого проекту, важливо зрозуміти необхідність впровадження протоколу IPv6 [1].

В сучасному світі, де відносно кожен аспект нашого життя стає все більш підключеним до Інтернету, зростає потреба в розширенні адресного простору Інтернету. Інтернет-протокол IPv4, який був введений більше 30 років тому, використовує 32-бітні IP-адреси, що обмежує кількість унікальних адрес до близько 4,3 мільярда [2]. З плином часу цей обмежений адресний простір став недостатнім для забезпечення зростаючої кількості підключених до Інтернету пристроїв.

Тому, з метою вирішення цієї проблеми та підтримки нових технологій та пристроїв, було розроблено IPv6 - наступну версію протоколу Інтернет-протоколу. IPv6 використовує 128-бітні IP-адреси, що забезпечує практично безкінечний адресний простір, оцінюється близько 3,4 * 10^38 унікальних адрес. Це дозволяє виділяти адреси не лише кожному підключеному до Інтернету пристрою, але й іншим об'єктам, які потребують з'єднання, таким як розумні прилади, датчики, автомобілі та багато іншого.

У зв'язку з цим, розробка графічного інтерфейсу для налаштування IPv6 адресації у локальній мережі Ethernet стає актуальною та важливою задачею [3]. Цей інтерфейс дозволить користувачам зручно та ефективно налаштовувати IPv6 параметри в мережі, сприяючи розширенню та покращенню зв'язку в Інтернеті на основі нового протоколу.

Дана робота актуальна сьогодні як ніколи. Так вийшло через те що IPv4 не може покрити кожний пристрій, хоча є NAT, який подовжує життя даного протоколу і наразі використовується, його метод закладається в тому що під

одною глобальною адресою можуть бути використані декілька адрес в локальній мережі, але це скоро не буде актуальним через появу більш ємного протоколу IPv6 який дозволить в теорії надавати на одну людину близько 300млн адрес [4]. У порівнянні з IPv4, який дає близько 0,5 адрес на одну людину, тобто ємність IPv6 у 1028 разів більша.

Метою є розробка та впровадження графічного інтерфейсу, який забезпечить зручний доступ до налаштування IPv6 адресації та відповідних параметрів в локальній мережі Ethernet, сприяючи покращенню управління мережевими настройками та забезпеченню безперебійного з'єднання між пристроями у сучасному цифровому світі.

1.2 Аналіз роботи IPv6

IPv6 (Internet Protocol version 6) є наступною версією протоколу Інтернет-протоколу, розробленою з метою вирішення проблем, що виникали зі зростанням кількості підключених до Інтернету пристроїв та обмеженим адресним простором IPv4.

Шоста версія протоколу IP, яка була випущена у 1996 році, є найсучаснішою на сьогоднішній день версією [5]. Ця версія відрізняється від попередніх в більшій цілісності передачі даних, швидкості та має інші значні переваги. Потреба у шостій версії випливає з простого факту - IP-адреси вичерпані. Це означає, що обсяг доступних адрес став недостатнім для покриття всіх пристроїв у мережі, включаючи комп'ютери, маршрутизатори та інші вузли. Вже у 80-х роках минулого століття експерти почали зауважувати, що IP-адрес не вистачає. Швидкий розподіл доступного адресного простору викликав проблему навантаження на внутрішні структури та інструкції. З кожним днем з'являлось все більше пристроїв, а Інтернет став необхідною складовою повсякденного життя. З цієї причини була введена класова адресація. Вона полягала у поділі на 5 класів - A, B і C для окремих вузлів, D для мультикастової передачі та E для експериментів [6].

Ось детальний аналіз роботи IPv6.

- Розширений адресний простір: Головною перевагою IPv6 є значне збільшення адресного простору. Замість 32-бітних адрес IPv4, IPv6 використовує 128-бітні адреси, що дозволяє створювати надзвичайно велику кількість унікальних адрес. Запас адресного простору IPv6 є практично нескінченним, що вирішує проблему нестачі адрес в IPv4.
- Покращена безпека: IPv6 включає в себе покращені механізми безпеки порівняно з IPv4 [7]. Вбудована підтримка IPsec у протоколі IPv6 забезпечує шифрування, аутентифікацію та цілісність даних, що пересилаються через мережу. Це забезпечує вищий рівень конфіденційності та безпеки мережевого зв'язку.
- Автоматична конфігурація: IPv6 включає механізми автоматичної конфігурації, які спрощують процес надання IP-адрес пристроям у мережі. Один з таких механізмів - Stateless Address Autoconfiguration (SLAAC), дозволяє пристроям отримувати унікальні IP-адреси без необхідності централізованого сервера DHCP. Це забезпечує швидку та ефективну настройку мережевих параметрів.
- Підтримка нових технологій та пристроїв: IPv6 було розроблено з урахуванням майбутніх потреб та змін у технологічному ландшафті. Він підтримує нові функції, які дозволяють підключати до Інтернету розумні прилади, IoT-пристрої, автомобілі, смарт-доми та інші пристрої, що розширює можливості підключення та забезпечує інтернет-зв'язок у різних сферах життя.
- Перехід від IPv4 до IPv6: У зв'язку з розширенням IPv6, проводяться зусилля для поступового переходу від IPv4 до IPv6. Цей процес є поступовим та потребує спільних зусиль від постачальників Інтернет-послуг, мережевих адміністраторів та виробників пристроїв.

Безумовно, всі перераховані вище зміни мають великий позитивний вплив на користувачів. Ось лише основні переваги IPv6 порівняно з IPv4.

- Значно покращена швидкість завантаження файлів: Раніше IPv4 використовувався з технологією перетворення адрес NAT, що дозволяло Інтернету працювати швидше. Проте, навіть порівнюючи IPv4 з NAT і IPv6, остання значно переважає в швидкості. Завдяки цьому, файли можна завантажувати набагато швидше.
- Покращена швидкість роботи з торрентами: Навіть якщо використовувати торрент-клієнти для завантаження файлів замість прямого завантаження через браузер, IPv6 забезпечує значний прогрес у швидкості [8]. Всі звичайні та незвичайні способи завантаження файлів, включаючи різні завантажувачі та клієнти, стають набагато швидшими.
- Покращена надійність: Крім швидкості, надійність є важливим фактором для користувачів, особливо при завантаженні потокового відео, файлів або перегляді сторінок у браузері. IPv6 дозволяє позбавитися від різних помилок, затримок та зависань, що забезпечує значне покращення цих аспектів.

Таким чином, перехід на IPv6 має великі переваги для користувачів, включаючи високу швидкість завантаження файлів, покращену швидкість роботи з торрентами та підвищену надійність [6].

Узагальнюючи, IPv6 є важливим кроком у розвитку Інтернету, забезпечуючи більшу кількість унікальних адрес, покращену безпеку та підтримку нових технологій. Перехід до IPv6 є необхідним для забезпечення масштабованого та стійкого розвитку мережевих зв'язків.

1.3 Аналіз аудиторії

Основна аудиторія, до якої буде спрямовано графічний інтерфейс налаштування IPv6 адресації у локальній мережі Ethernet, - це люди, які не мають глибоких знань та розуміння процесу налаштування IPv6 адресів. Завдання полягає в тому, щоб надати їм простий, зрозумілий та інтуїтивно зрозумілий

інтерфейс, який дозволить їм без зусиль налаштовувати свою мережу з використанням IPv6.

Оскільки користувачі не мають спеціалізованого знання про налаштування IPv6 адресів, планується створити інтерфейс, що не потребує глибоких технічних знань. В проекті будуть використовуватись прості терміни та поняття, які зрозумілі для широкого кола користувачів. Крім того, буде включено довідкову інформацію та пояснення для кожного поля та налаштування, щоб допомогти користувачам зрозуміти, які опції вони мають та як їх правильно використовувати.

Метою є забезпечити максимальну простоту використання. Буде створено інтерфейс з інтуїтивним дизайном, де налаштування та опції будуть логічно розташовані та легко доступні. Кнопки та елементи управління будуть ясні та прості, щоб користувачі могли з легкістю здійснювати потрібні дії.

Крім того, для полегшення використання інтерфейсу, буде реалізовано механізми перевірки правильності введених даних та відображення попереджень або помилок у разі некоректних введень. Це допоможе уникнути можливих помилок та забезпечить користувачам відчуття впевненості у своїх налаштуваннях.

Загалом, графічний інтерфейс буде спрямований на людей без спеціалізованого розуміння налаштування IPv6 адресації. Буде створено інтуїтивно зрозумілий та простий інтерфейс, щоб допомогти користувачам без зусиль налаштовувати свою мережу з використанням IPv6.

1.4 Інструменти для проектування схеми

Для виконання роботи потрібна програма щоб наглядно показати принцип роботи IPv6. Такою програмою стала Cisco Packet Tracer. Для кого Cisco Packet Tracer? В першу чергу це учні, націлені на кар'єру у сфері IT та мережевих технологій, студенти, які проходять навчання в галузі мережевих технологій,

інтернету речей та кібербезпеки, інженери, викладачі та інструктори та для дистанційного навчання [9].

Cisco Packet Tracer - це програмне забезпечення, розроблене компанією Cisco Systems, призначене для моделювання, налагодження і тестування мережевих конфігурацій [10]. Воно дозволяє студентам, викладачам і мережевим інженерам експериментувати з різними мережевими сценаріями, створювати віртуальні мережі та спостерігати за їх роботою без фізичного обладнання. Основні приклади використання Cisco Packet Tracer.

- Навчання мережевих технологій: Packet Tracer є популярним інструментом для навчання студентів мережевим технологіям. Він дозволяє створювати віртуальні мережі і експериментувати з різними конфігураціями без необхідності фізичного обладнання. Це допомагає студентам отримати практичні навички в роботі з мережами і виконати вправи та завдання, пов'язані з маршрутизацією, комутацією, настроюванням мережевих пристроїв та іншими аспектами мережевої інфраструктури.
- Дослідження мережевих сценаріїв: Cisco Packet Tracer дозволяє створювати складні мережеві сценарії з різними типами мережевих пристроїв, такими як маршрутизатори, комутатори, сервери, мережеві мережеві пристрої та інші. Це дозволяє мережевим інженерам проводити дослідження, моделювати і тестувати різні конфігурації мережі, перевіряти сумісність і пропускну здатність, виявляти і вирішувати проблеми мережі та вдосконалювати її дизайн.
- Візуалізація мережевих концепцій: Packet Tracer надає графічний інтерфейс, що дозволяє створювати візуальні моделі мереж, що допомагає в розумінні мережевих концепцій і взаємодії між пристроями. Візуалізація допомагає відображати топологію мережі, маршрутизацію, комутацію трафіку і багато іншого, що допомагає краще розуміти принципи мережевого дизайну та роботу пристроїв.

– Підготовка до сертифікаційних випробувань: Packet Tracer може бути використаний для підготовки до сертифікаційних випробувань, таких як сертифікація Cisco CCNA. Він дозволяє студентам і претендентам на сертифікацію практикувати і вдосконалювати свої навички в настроюванні мережевих пристроїв, вирішуванні проблем та інших завданнях, які вимагаються для успішної сертифікації.

Cisco Packet Tracer є потужним інструментом для моделювання мереж та експериментів з мережевими конфігураціями. Він дозволяє студентам і мережевим інженерам навчатися, досліджувати та вдосконалювати свої навички в галузі мережевих технологій.

Для моєї роботи я використовую програму націлену на студенті, її цілком достатньо.

1.5 Інструменти для створення графічного інтерфейсу

Планується створення графічного інтерфейсу на мові програмування C# для налаштування IPv6 адресації у локальній мережі Ethernet [11]. Проект буде забезпечувати зручний та інтуїтивно зрозумілий інтерфейс, що дозволить користувачам ефективно керувати та контролювати свої мережеві параметри.

Основною метою проекту буде надання можливості вводу та зміни IPv6 адрес у локальній мережі Ethernet [12]. Буде забезпечено все необхідне, щоб користувач міг вводити адреси в зручний формат. Будуть враховані особливості IPv6 адресації, такі як використання двокрапки та шістнадцяткових чисел. Крім того, планується реалізувати функцію автоматичного визначення IPv6 адреси, яка дозволить користувачу автоматично отримати IP-адресу від мережевого пристрою.

Додатково, графічний інтерфейс буде містити можливості налаштування мережевих параметрів, таких як шлюз за замовчуванням та IP для комп’ютерів та маска для них. Це дозволить користувачам налаштувати мережеве з'єднання згідно з їх потребами. Інформація про поточні налаштування буде відображатися

у режимі реального часу, щоб користувачі могли відстежувати стан свого підключення.

Виконане це буде з використанням мови програмування C# та з використанням платформи .NET, що забезпечить мені потужність і гнучкість для розробки потрібного інтерфейсу [13]. Під час розробки буде дотримано кращих практик програмування, таких як чистий код та модульна структура, що забезпечить підтримку та розширення проекту в майбутньому.

Реалізовано цей код буде за допомогою Microsoft Visual Studio (рис. 1.1).

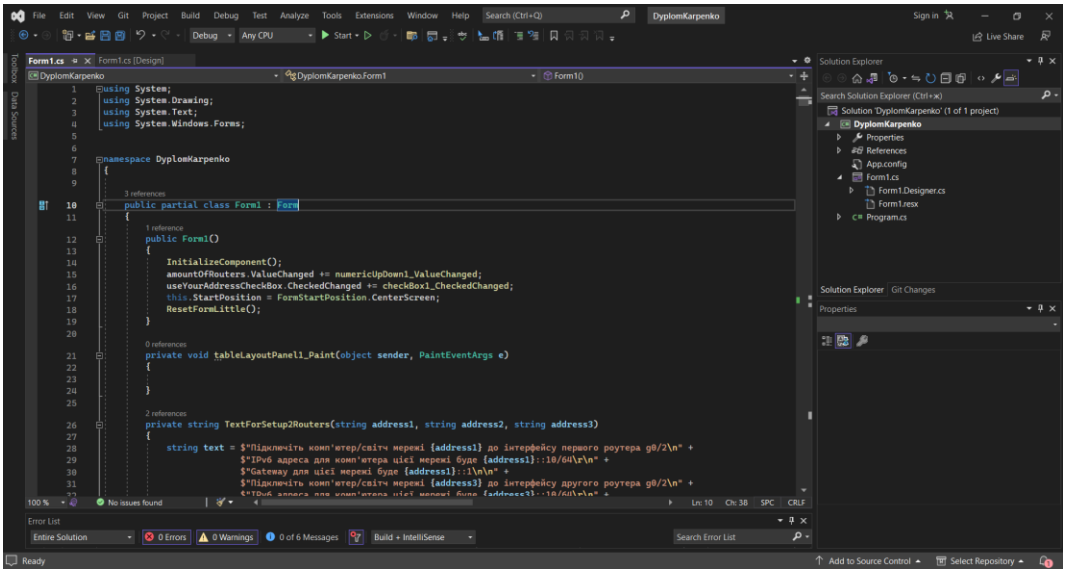

Рисунок 1.1 – Інтерфейс Microsoft Visual Studio

Visual Studio - це інтегроване середовище розробки (Integrated Development Environment, IDE), створене компанією Microsoft, призначене для розробки програмного забезпечення [14]. Воно надає розширений набір інструментів і функцій, які допомагають програмістам писати, налагоджувати і тестувати програми на різних платформах. Visual Studio спеціально розроблено для підтримки роботи з мовами програмування, такими як C#, C++, Visual Basic .NET, F# та іншими [15]. Інтегровані засоби підтримують автоматичне вирішення проблем залежностей, підказки інтелектуального коду, налагодження, відстеження помилок, тестування і багато інших функціональних можливостей. Основні переваги та можливості Visual Studio для мови програмування C#.

- Розширена підтримка мови C#: Visual Studio надає повний спектр засобів для розробки програм на мові C#. Включаючи інтегровану підказку коду, автоматичне завершення коду, підтримку відступів, рефакторинг, аналіз коду та багато іншого.
- Інструменти налагодження: Visual Studio має потужні засоби для налагодження програм. Він дозволяє крокувати по коду, встановлювати точки зупину, спостерігати за значеннями змінних, аналізувати стек викликів і виконувати багато інших операцій для виявлення та виправлення помилок.
- Інтеграція зі засобами контролю версій: Visual Studio підтримує інтеграцію з популярними системами контролю версій, такими як Git. Це дозволяє зручно відслідковувати зміни в коді, працювати зі спільними проектами та співпрацювати з іншими розробниками.
- Широкий вибір інструментів і розширень: Visual Studio має велику кількість доступних інструментів і розширень, які можна використовувати для поліпшення продуктивності і розширення функціональності розробки. Це можуть бути додаткові плагіни, шаблони проектів, засоби тестування, пакети пакування та багато інших.
- Можливість розробки для різних платформ: Visual Studio підтримує розробку програмного забезпечення для різних платформ, включаючи Windows, веб, мобільні пристрої і хмарні сервіси. Це дозволяє розробникам створювати різноманітні застосунки, забезпечувати кросплатформенність і розширювати свою аудиторію.

Visual Studio є потужним інструментом для розробки програмного забезпечення, забезпечуючи програмістам зручне середовище для написання, тестування і налагодження коду. Він підтримує широкий спектр мов програмування та має багато корисних функцій і розширень, що допомагають

покращити продуктивність розробки в тому числі розробка графічного інтерфейсу (рис. 1.2).

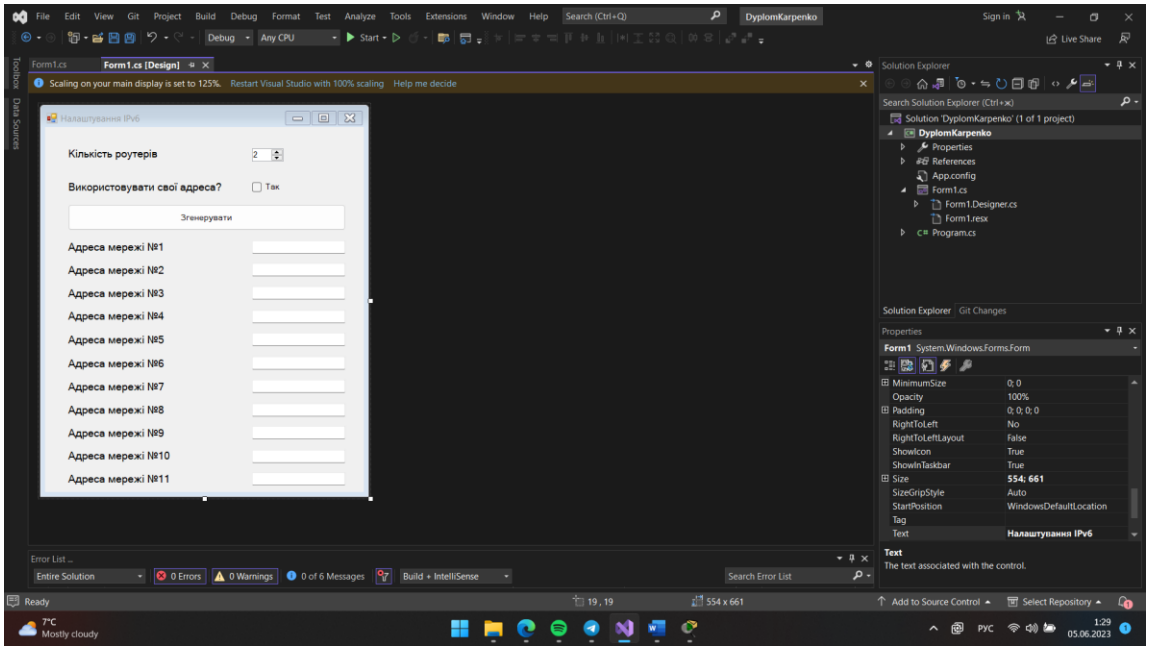

Рисунок 1.2 – Інтерфейс розробки графічного інтерфейсу Visual Studio

У підсумку, проект буде надавати зручну можливість керувати мережевими налаштуваннями, що дозволить користувачам легко налаштовувати їх мережу з IPv6 адресацією.

1.6 Постановка задачі

Для того щоб показати як буде працювати графічний інтерфейс необхідно реалізувати наступні задачі:

- виконати огляд інструменту для створення схем та зрозуміти як створити схему;
- створити базову схему без налаштувань;
- підписати схему;
- виконати налаштування IPv6;
- виконати тестування схеми та зробити відповідні висновки по роботі;
- спроектувати вигляд графічного інтерфейсу;

- відтворити вигляд графічного інтерфейсу за допомогою Microsoft Visual Studio, а саме Windows Forms;
- написати програму яка буде виводити файл з необхідними даними;

У разі успішної реалізації при введені даних з файлу в роутери схема повинена працювати, пакети даних повинні доставлятись.

2 ВИБІР МЕТОДІВ РОЗВ'ЯЗАННЯ ЗАДАЧІ

2.1 Огляд інструменту Cisco Packet Tracer

За період навчання в СумДУ було пройдено курс CCNA тому вже було набуто базові навички з роботою у Packet Tracer (рис. 2.1) через що проблем з роботою в даній програмі не було [16].

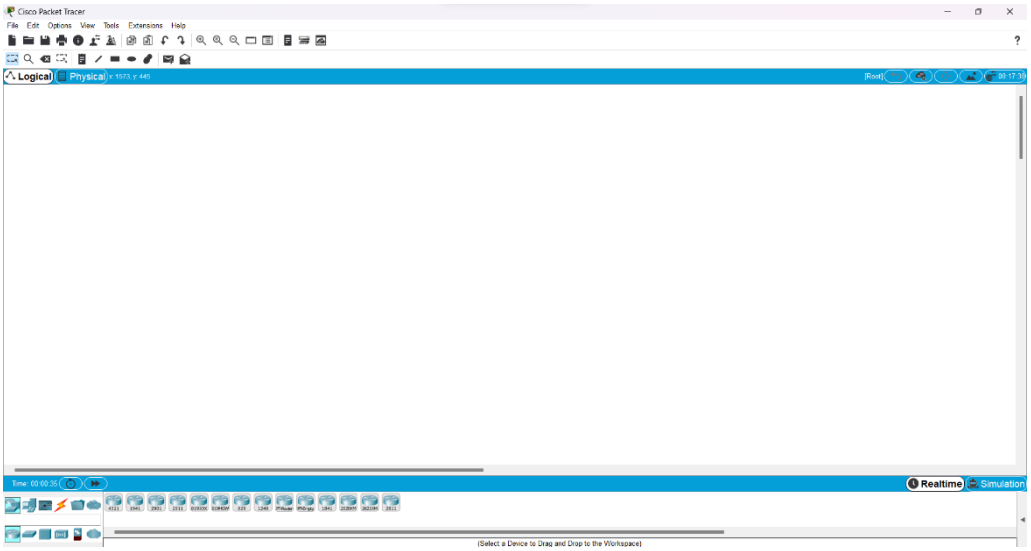

Рисунок 2.1 – Інтерфейс Cisco Packet Tracer

Дана програма призначена для симулювання реальних випадків застосування телекомунікаційних пристроїв. Cisco Packet Tracer має великий набір маршрутизаторів, світчів, варіантів з'єднань, та іншого. Тобто ця програма гарна для демонстрації роботи протоколів, що мені і потрібно.

2.2 Огляд інструменту Microsoft Visual Studio

У продовж навчання в СумДУ часто зустрічались С подібні мови програмування тому була обрана мова програмування C# для реалізації своєї програми. Через це було обрано відповідну IDE Visual Studio від Microsoft.

Дана програма ідеально підійшла для реалізації проекту через наявність Windows Forms яка має деякі переваги (табл. 2.1) над іншими аналогами.

Таблиця 2.1 – Переваги Windows Forms [17]

Назва	Опис
Простота використання	Windows Forms відомі своєю простотою в розробці. Вони базуються на класичній моделі програмування, де інтерфейс користувача створюється шляхом розташування контролів на формі та обробки подій. Це дозволяє швидко створювати прості та середньої складності додатки з мінімальними зусиллями.
Багатофункціональність	Windows Forms надають широкий набір контролів, які можна використовувати для створення багатофункціонального інтерфейсу. Вони включають кнопки, текстові поля, списки, таблиці, вкладки та багато інших. Це дозволяє розробникам швидко створювати різноманітні елементи керування та взаємодіяти з користувачем.
Широке сприйняття	Windows Forms мають широке сприйняття серед розробників. Вони існують протягом тривалого часу і мають велику базу знань та ресурсів. Це означає, що розробники можуть легко знайти документацію, приклади та рішення проблем, пов'язаних з Windows Forms, що сприяє швидкому та ефективному розвитку додатків.
Інтеграція зі старішими системами	Windows Forms добре підходять для інтеграції зі старішими системами та платформами. Якщо вам потрібно розробити додаток, який взаємодіє зі старішою версією Windows або іншими програмами, Windows Forms можуть бути більш придатним варіантом, оскільки вони мають велику сумісність з попередніми версіями Windows та бібліотеками класів.

Windows Forms є ефективною технологією для розробки класичних настільних додатків під Windows. Вони прості у використанні, мають широку підтримку та можуть бути ефективним рішенням для багатьох проектів.

2.3 Проектування вигляду програми

Вхідні дані мого проекту досить прості через те що поєднуватись роутери будуть послідовно. Тобто це будуть такі дані які буде вказувати користувач.

- Кількість роутерів (від 2 до 10).
- Автоматично надавати адреси (Так, ні).
- Якщо ні, користувач вписує IPv6 для кожної з мереж (кількість мереж залежить від кількості роутерів).

Отже вигляд інтерфейсу буде таким якщо користувач вирішить не вписувати IPv6 (рис. 2.2):

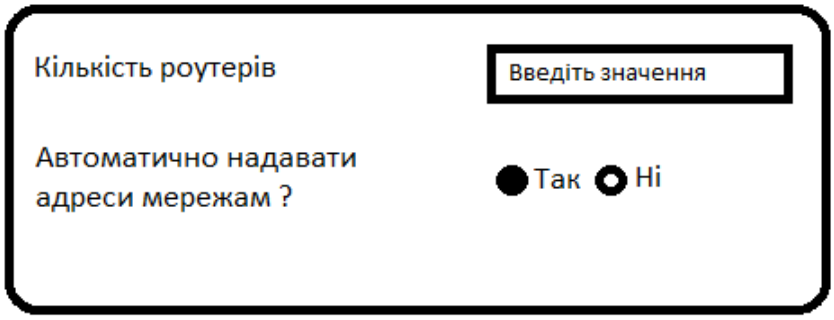

Рисунок 2.2 – Інтерфейс з автоматичним надаванням адрес

Проте якщо користувач захоче ввести свої адреси інтерфейс буде виглядати трохи інакше (рис. 2.3):

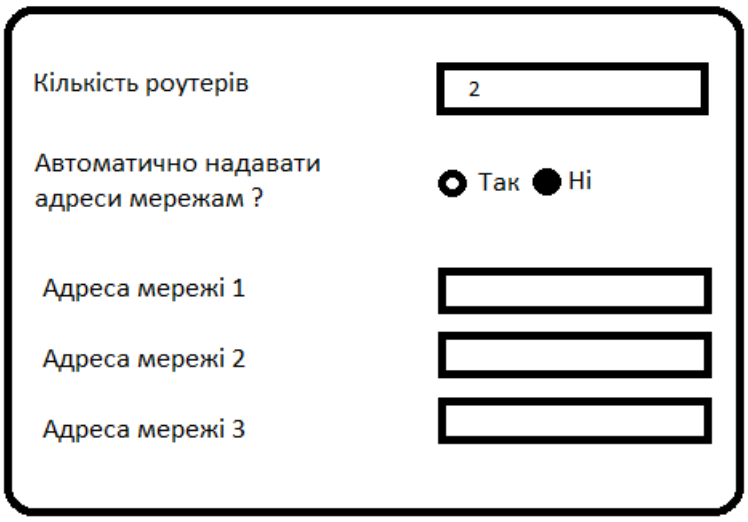

Рисунок 2.3 – Інтерфейс без автоматичного надавання адрес

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка дизайну програми

З початку треба додати кнопку взаємодії користувача з програмою для генерації вихідного файлу в якому будуть створені команди для роутерів. Процес додавання кнопок в Windows Forms з використанням мови програмування C# є необхідним етапом розробки інтерфейсу користувача. Для досягнення цієї мети слід виконати наступні кроки.

– У вікні дизайнера форми треба виконати процес візуального моделювання інтерфейсу користувача, де можна відображати та змінювати елементи.

– За допомогою інструменту "Button" на панелі інструментів (рис. 3.1) (зазвичай зліва або вгорі вікна дизайнера форми) треба додати кнопку на форму, просто клацнувши на ній.

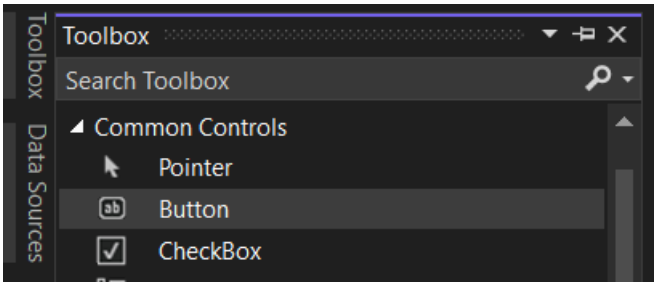

Рисунок 3.1 – Панель інстурментів де знаходиться "Button"

– Можна використовувати панель властивостей (рис. 3.2) (зазвичай знаходиться справа вікна дизайнера форми) для налаштування параметрів кнопки, таких як текст, розмір, положення та стиль.

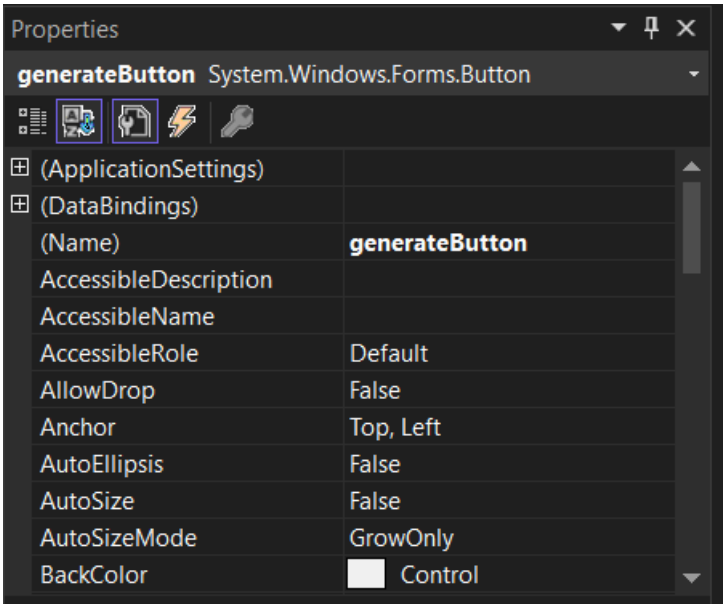

Рисунок 3.2 – Панель властивостей "Button"

– Для задання дії, що виконується при натисканні кнопки, подвійно потрібно клацнути на ній, щоб створити обробник події "Click". Це автоматично відкриє кодовий редактор зі зразком функції, яку можна налаштувати за потреби.
– У кодовому редакторі визначити логіку, яка виконується під час натискання кнопки, наприклад, відкриття нового вікна, виконання обчислень або збереження даних.
– Запустити програму, щоб переглянути інтерфейс користувача та переконайтись, що кнопки працюють відповідно до очікувань (рис. 3.3).

Важливо враховувати, що конкретні деталі та параметри можуть варіюватися в залежності від версії Visual Studio або використовуваного інтегрованого середовища розробки.

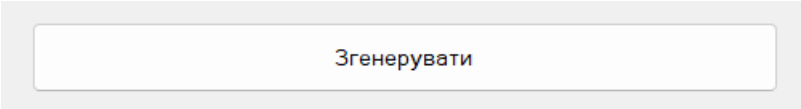

Рисунок 3.3 – Остаточний вигляд кнопки "Button"

Потім потрібно було додати CheckBox для перевірки чи хоче користувач використовувати свої адреси, або згенерувати їх.

Спочатку зроблено підпис для цього CheckBox. Створено його методом додавання Label. Для реалізації цього процесу слід дотриматись наступних кроків.

– У вікні дизайнера форми, візуально розмістити та налаштувати елементи інтерфейсу користувача.
– За допомогою інструменту "Label" (мітка) на панелі інструментів (рис. 3.4) (зазвичай зліва або вгорі вікна дизайнера форми) додати мітку на форму, просто клацнувши на потрібному місці.

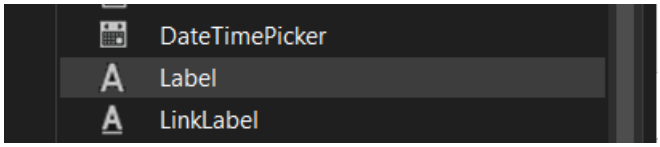

Рисунок 3.4 – Панель інстурментів де знаходиться "Label"

– З використанням панелі властивостей (рис. 3.5) (зазвичай знаходиться справа вікна дизайнера форми) налаштувати параметри мітки. Наприклад, можна змінити текст, розмір, положення, кольори та шрифт мітки.

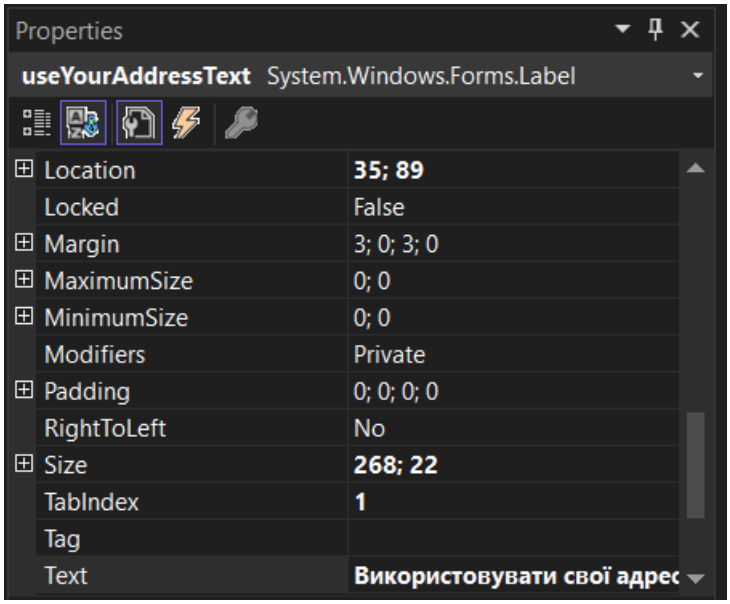

Рисунок 3.5 – Панель властивостей "Label"

– Після встановлення відповідних властивостей мітки, перейти до режиму редагування коду, подвійно клацнувши на мітці. Це відкриє кодовий редактор з генерованою функцією, яку можна налаштувати за своїми потребами.
– В кодовому редакторі визначити логіку, пов'язану з міткою, наприклад, обробку подій або динамічну зміну вмісту мітки відповідно до потреб.
– Запустити програму, щоб переглянути інтерфейс користувача та переконатися, що мітки відображаються вірно і виконують необхідні функції, відповідно до вимог (рис. 3.6).

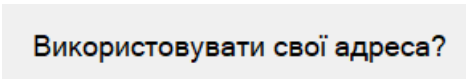

Рисунок 3.6 – Остаточний вигляд напису "Label"

Після створення підпису можна додати і сам CheckBox. Зробити це можно дотримуючись таких дій.

– Відкрити вікно дизайнера форми, де буде розміщено та налаштовано елементи інтерфейсу користувача.

– Знайти елемент CheckBox на панелі інструментів (рис. 3.7). Елемент CheckBox зазвичай позначається як прапорець або квадратик з текстом.

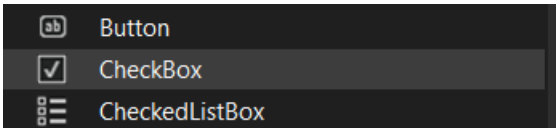

Рисунок 3.7 – Панель інстурментів де знаходиться "CheckBox"

– Додати елемент CheckBox на форму, натиснувши на нього або перетягнувши його на потрібне місце на формі.
– За допомогою панелі властивостей (рис. 3.8) налаштувати параметри елемента CheckBox. Наприклад, можна змінити текст, шрифт, колір фону та тексту елемента CheckBox.

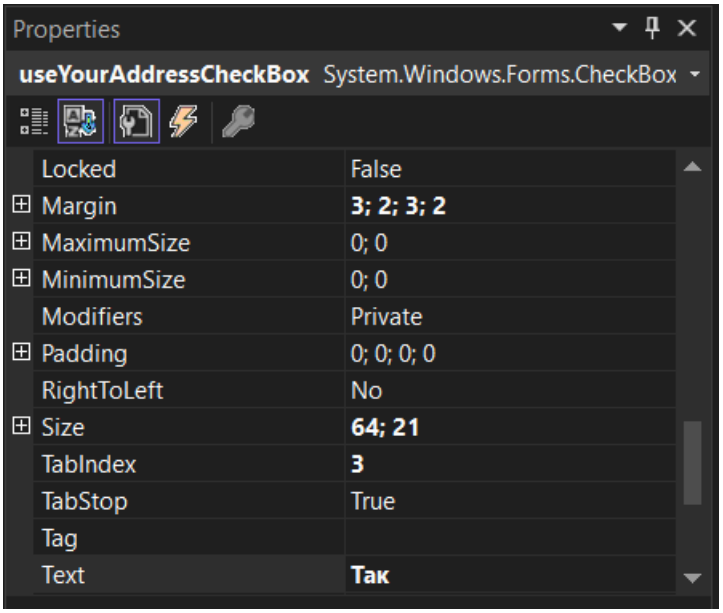

Рисунок 3.8 – Панель властивостей "CheckBox "

– В кодовому редакторі визначити логіку, пов'язану з елементом CheckBox, якщо вона необхідна. Наприклад, можливо обробити події зміни стану CheckBox або зчитувати/змінювати його значення в коді програми.
– Перевірити, чи працює елемент CheckBox, запустивши програму і

спробувавши клікнути на нього, щоб змінити його стан.

Тепер потрібен NumericUpDown для того щоб користувач міг задати кількість роутерів для яких будуть згенеровані адреси, попередньо треба додати напис за допомогою Label процес додавання якого описано вище. Ось процес додавання NumericUpDown в Windows Forms.

– У вікні дизайнера форми, візуально розмістити та налаштувати елементи інтерфейсу користувача.
– Знайдти елемент NumericUpDown у панелі інструментів (рис. 3.9) (зазвичай знаходиться зліва або вгорі вікна дизайнера форми). Він зазвичай позначається як стрілочки вгору і вниз поряд з полем введення числа.

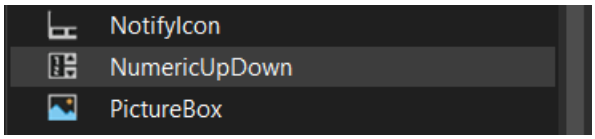

Рисунок 3.9 – Панель інструментів де знаходиться "NumericUpDown"

– Додати елемент NumericUpDown на форму, просто клацнувши на ньому або перетягуючи його на потрібне місце на формі.
– З використанням панелі властивостей (рис. 3.10) (зазвичай знаходиться справа вікна дизайнера форми) налаштувати параметри елемента NumericUpDown. Наприклад, встановити початкове значення, мінімальне і максимальне значення, інтервал збільшення або зменшення, форматування числа та інші властивості, які відповідають потребам.

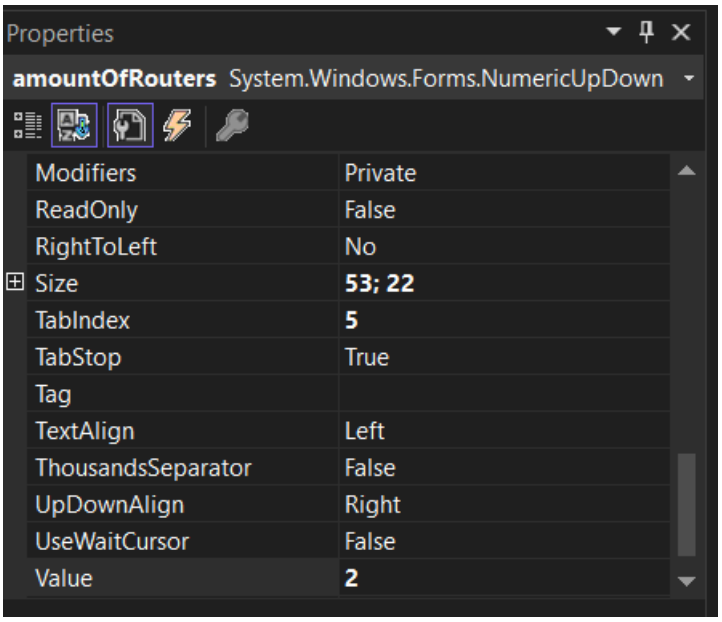

Рисунок 3.10 – Панель властивостей "NumericUpDown"

– Перевірити, чи працює елемент NumericUpDown, запустивши програму і перевіривши його функціональність. Користувач повинен мати змогу встановити бажану кількість роутерів за допомогою елемента NumericUpDown (рис. 3.11).

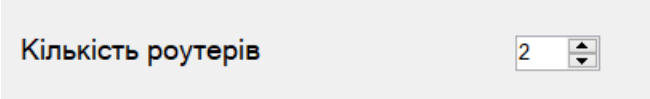

Рисунок 3.11 – Остаточний вигляд "NumericUpDown"

Тепер потрібно додати TextBox для того щоб користувач міг вводити свої IPv6 адреси, їх потрібно одинадцять штук, через те що в проекті максимальна кількість роутерів шість штук. Додавання цих елементів відбувається таким чином.

– Відкрити вікно дизайнера форми, де треба розміщувати та налаштовувати елементи інтерфейсу користувача.
– Знайти елемент TextBox на панелі інструментів (рис. 3.12). Елемент TextBox зазвичай позначається як текстове поле для введення даних.

Рисунок 3.12 – Панель інструментів де знаходиться "TextBox"

– Додати елемент TextBox на форму, натиснувши на нього або перетягнувши його на потрібне місце на формі.
– Використовуючи панель властивостей (рис. 3.13), треба налаштувати параметри елемента TextBox. Наприклад, можна змінити розмір, положення, шрифт, колір фону та тексту елемента TextBox.

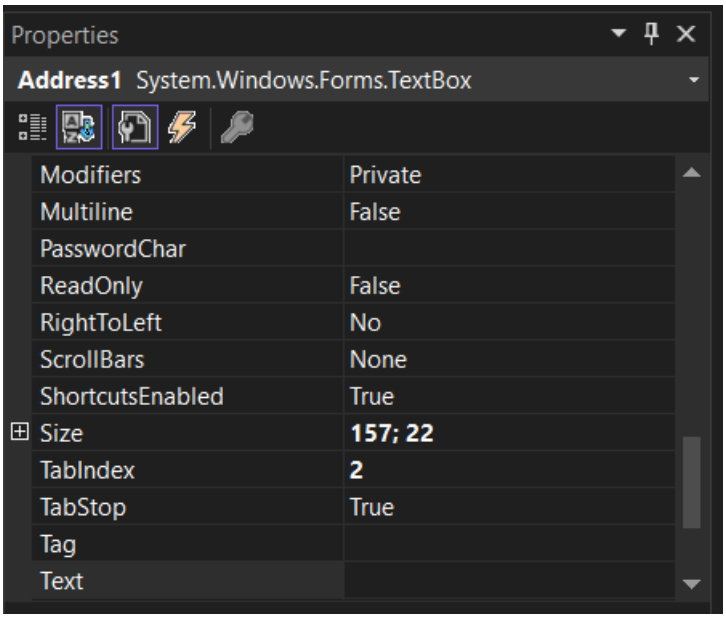

Рисунок 3.13 – Панель властивостей "TextBox"

– В кодовому редакторі визначити логіку, пов'язану з елементом TextBox. Наприклад, можливо обробити події зміни тексту, натискання клавіш або фокусування на елементі TextBox.
– Перевірити, чи працює елемент TextBox, запустивши програму і спробувавши вводити та редагувати текст у полі TextBox.

Також потрібно додати до цих текстових форм підписи (рис. 3.14) за допомогою Label, процес якого описаний вище.

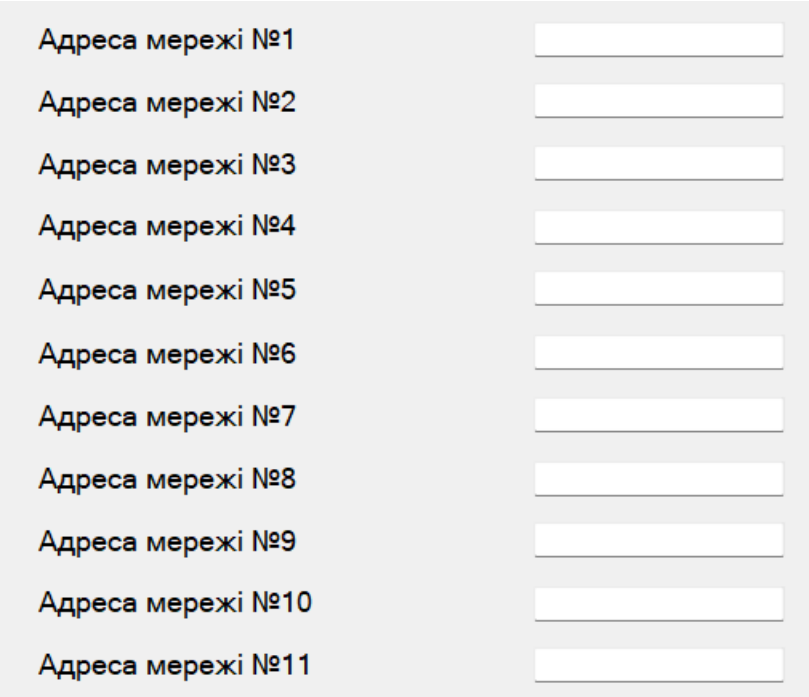

Рисунок 3.14 – Остаточний вигляд "TextBox" з підписами

Після виконаних дій створено остаточний вигляд програми (рис. 3.15).

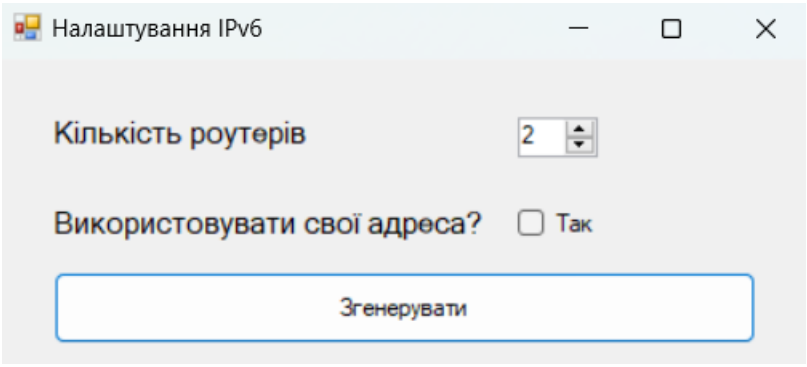

Рисунок 3.15 – Вигляд дизайну форми з генерацією адрес

Це варіант вигляду (рис. 3.15) для того варіанту коли користувач не бажає використовувати свої адреси. Якщо користувач захоче використовувати свої адреси то вигляд буде таким (рис 3.16):

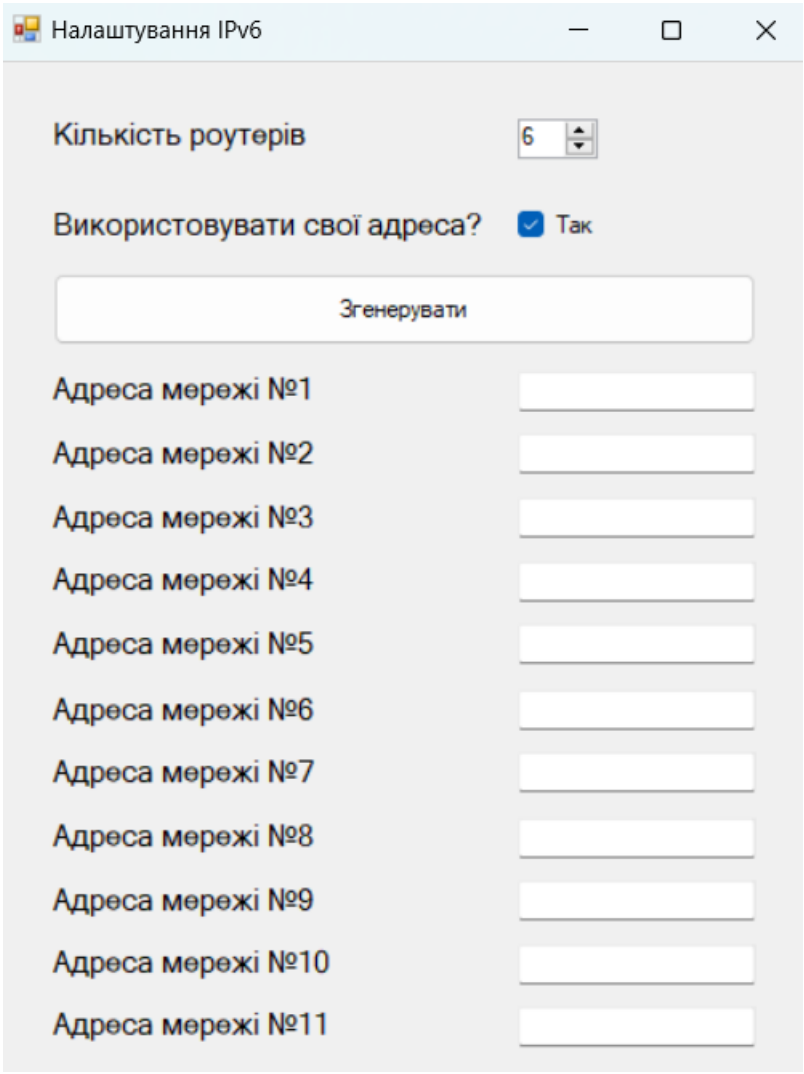

Рисунок 3.16 – Вигляд дизайну форми без генерації адрес

Також вигляд змінюється в залежності від обраної кількості роутерів, ось приклад вигляду для чотирьох роутерів (рис. 3.17):

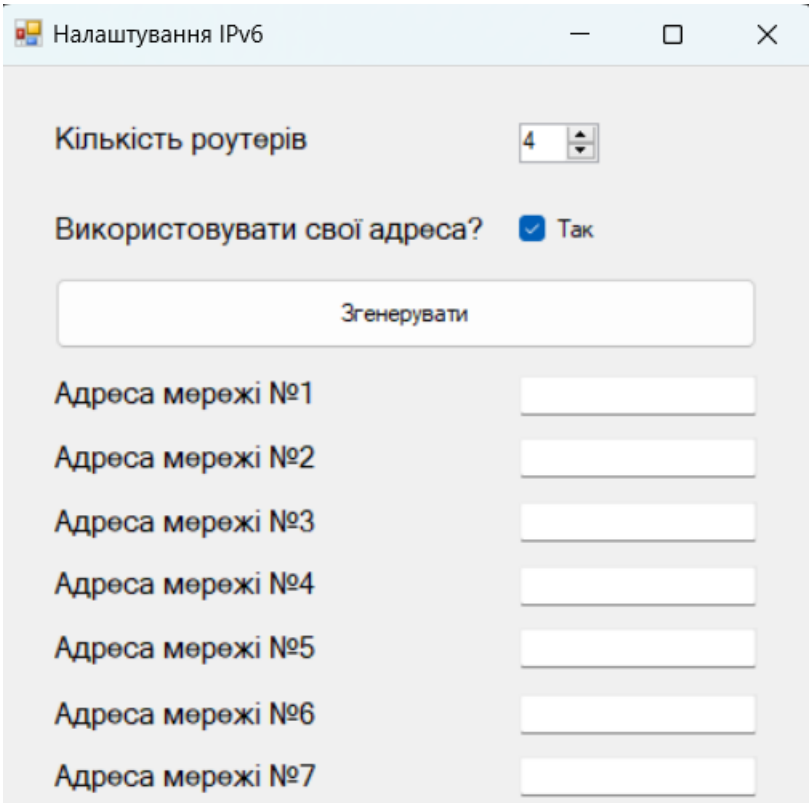

Рисунок 3.17 – Вигляд дизайну форми для чотирьох роутерів без генерації адрес

Зроблено це за допомоги програмної реалізації, а точніше за допомогою атрибуту visible. Для цього створено окрему функцію UpdateAddressVisibility (рис. 3.18) (Додаток А). У методі UpdateAddressVisibility перевіряється, чи встановлено прапорець useYourAddressCheckBox.Checked, що вказує на використання власної адреси. Якщо так, то в залежності від значення в полі amountOfRouters (кількість маршрутизаторів) виконуються різні гілки умовного оператора if-else. У кожній гілці встановлюється видимість полів адреси (Address1, Address2, тощо) та текстових підписів (Address1Text, Address2Text, тощо) відповідно до кількості маршрутизаторів. Також змінюється розмір вікна форми (this.Size) для відображення додаткових полів адреси. Якщо значення amountOfRouters не відповідає жодному з визначених умов, викликається метод ResetFormLittle (Додаток Б), що скидає форму до початкового стану. Якщо прапорець useYourAddressCheckBox.Checked не встановлено, також викликається метод ResetFormLittle (Додаток Б), що скидає форму до початкового стану.

```
private void UpdateAddressVisibility()
{
    if (useYourAddressCheckBox.Checked)
    {
        int value = (int)amountOfRouters.Value;

        if (value == 2)
        {
            ResetFormLittle();
            Address1.Visible = true;
            Address1Text.Visible = true;
            Address2.Visible = true;
            Address2Text.Visible = true;
            Address3.Visible = true;
            Address3Text.Visible = true;
            this.Size = new Size(425, 286);
        }
```

Рисунок 3.18 – Вигляд коду для двох роутерів

3.2 Розробка генерації вихідного файлу

Для того щоб користувач зміг скористуватись згенерованим кодом для вводу в консоль роутерів було обрано рішення генерувати файл з необхідним текстом. З початку було описано код для роутерів який буде записуватись в вихідний файл. Приклад такого тексту описано в Додатку В. Цей текст було описано для кожної ситуації, тобто окремо було створено текст для двух, троьох, чотирьох, п’яти та шести роутерів. Цей текст буде зберігатись в string, для цього було створені відповідні функції (рис. 3.19). Повний варіант функції описано в Додатку Г.

```
private string TextForSetup2Routers(string address1, string address2, string address3)
{
    string text = $"Підключіть комп'ютер/світч мережі {address1} до інтерфейсу першого роутера g0/2\n" +
                  $"IPv6 адреса для комп'ютера цієї мережі буде {address1}::10/64\r\n" +
                  $"Gateway для цієї мережі буде {address1}::1\n\n" +
                  $"Підключіть комп'ютер/світч мережі {address3} до інтерфейсу другого роутера g0/2\n" +
                  $"IPv6 адреса для комп'ютера цієї мережі буде {address3}::10/64\r\n" +
                  $"Gateway для цієї мережі буде {address3}::1\n\n" +
                  $"Підключіть кросовером перший роутер до другого роутера таким чином щоб на першому роутері
                  $"\n" +
```

Рисунок 3.19 – Функція що записує текст в string

У цій функції використовуються змінні addresss1, addresss2, addresss3. Вони передаються в цю функцію в методі button1_Click (Додаток В). Метод відповідає за обробку події натискання кнопки `button1`. У цьому коді

виконується перевірка значення `amountOfRouters.Value`, яке представляє кількість маршрутизаторів, вибрану користувачем. В залежності від цього значення виконується відповідна логіка. Якщо кількість маршрутизаторів дорівнює 2, то спочатку перевіряється, чи встановлено прапорець `useYourAddressCheckBox.Checked`. Якщо так, то зчитуються значення з текстових полів `Address1`, `Address2` і `Address3`. Якщо ці поля не порожні, то викликається функція `TextForSetup2Routers`, яка створює текстовий вихідний код для налаштування двох маршрутизаторів з використанням введених адрес. Цей текст записується у файл з ім'ям "output.txt", і після цього файл відкривається у відповідному застосунку. Також виводиться повідомлення про успішне створення і відкриття файлу. Якщо ж хоча б одне з полів має пусте значення, то виводиться повідомлення про помилку. Якщо прапорець `useYourAddressCheckBox.Checked` не встановлено, то генеруються випадкові значення для адрес маршрутизаторів за допомогою функції `GenerateIPv6Address`. Подальша логіка аналогічна попередній ситуації зі зчитуванням значень полів, створенням текстового вихідного коду, записом у файл, відкриттям файлу та виведенням повідомлення про успішне створення і відкриття файлу. Така сама логіка застосовується для решти варіантів кількості маршрутизаторів (3, 4, 5 і 6). Кожен варіант має свою функцію для створення текстового вихідного коду, яка приймає відповідну кількість аргументів. Якщо кількість маршрутизаторів не відповідає жодному з варіантів (2, 3, 4, 5 або 6), виводиться повідомлення про помилку. Загалом, цей код виконує створення файлу з налаштуваннями для мережі маршрутизаторів залежно від вибраної кількості маршрутизаторів та введених або згенерованих адрес.

Також в цій функції використовується генератор IPv6 адрес (рис. 3.20).

```
private Random random = new Random();

35 references
private string GenerateIPv6Address()
{
    StringBuilder sb = new StringBuilder();

    // Генерація сегментів IPv6-адрес
    for (int i = 0; i < 4; i++)
    {
        string segment = random.Next(0, 65536).ToString("X4");
        sb.Append(segment);

        // Додавання дільника ":" між сегментами
        if (i < 3)
        {
            sb.Append(":");
        }
    }

    return sb.ToString();
}
```

Рисунок 3.20 – Вигляд функції генератору IPv6 адрес

Розглянемо кожен рядок коду окремо.

– `private Random random = new Random();`

  Цей рядок створює новий об'єкт класу `Random`, який використовується для генерації випадкових чисел. Він ініціалізується заразованим значенням, щоб кожен раз, коли метод `GenerateIPv6Address` викликається, він використовував нову послідовність випадкових чисел.

– `StringBuilder sb = new StringBuilder();`

  Цей рядок створює об'єкт `StringBuilder`, який використовується для зберігання та збирання рядків. Він використовується для побудови IPv6-адресу крок за кроком.

– `for (int i = 0; i < 4; i++)`

  Цей цикл `for` виконується чотири рази, один раз для кожного сегмента IPv6-адресу. Кожен сегмент представляє дві шістнадцяткові цифри (біти).

- `string segment = random.Next(0, 65536).ToString("X4");`

  У цьому рядку генерується випадкове число в діапазоні від 0 до 65536 (включно). Число перетворюється на рядкове представлення за допомогою метода `ToString("X4")`. Формат "X4" вказує, що число повинно бути представлене у шістнадцятковому форматі з чотирма цифрами. Наприклад, число 10 буде представлене як "000A", а число 255 - як "00FF".

- `sb.Append(segment);`

  Цей рядок додає згенерований сегмент до об'єкту `StringBuilder`.

- `if (i < 3)`

  Цей рядок перевіряє, чи це не останній сегмент IPv6-адресу. Якщо це не останній сегмент, то наступний роздільник `":"` додається після сегмента.

- `return sb.ToString();`

  Після завершення циклу `for` зібраний IPv6-адрес у вигляді рядка повертається з методу.

Отже, цей метод генерує випадковий IPv6-адрес шляхом генерації чотирьох шістнадцяткових сегментів та їх об'єднання в рядок за допомогою роздільників ":". Наприклад, згенерований IPv6-адрес може мати вигляд "2001:0DB8:AC10:FE01".

ВИСНОВКИ

В процесі роботи створено графічний інтерфейс налаштування локальної мережі IPv6. Для виконання цієї роботи було виконано комплекс завдань. Проведено аналіз літератури, присвяченій принципам роботи IPv6, локальної мережі та порівняно протоколи IPv4 та IPv6, з метою постановки задачі для створення додатку. Розглянуто інструмент для створення схем локальних мереж. Проаналізовано інструмент для написання програми на мові C# з використанням Windows Forms. Розроблено дизайн графічного інтерфейсу додатку. Впроваджено дизайн додатку для легкого користування. Адаптовано дизайн під різні випадки використання програми у випадку зміни кількості роутерів. Створено генератор IPv6 адрес. Написно базовий текст який містить необхідні команди для роутерів. Реалізовано записування тексту в окрему змінну за допомогою функції. Написано код для різних випадків використання програми в залежності від кількості роутерів обраних користувачем. Зроблено генерацію вихідного файлу та відкриття його для перегляду користувачем.

Створена програма може бути розширена додатковим функціоналом. Наприклад, впровадження перевірки знаходження в текстовому полі саме адреси, а не будь якого тексту. Також функціонал можу бути розширений методом збільшення кількості роутерів для яких будуть генеруватись команди. Може бути впроваджене введення цих команд одразу в роутери без необхідного втручання користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IPv6: що це і навіщо [Electronic resource]. URL: https://senior.ua/articles/ipv6-scho-ce--navscho (accessed: 31.05.2023).
2. Що таке адреса IPv4? - techukraine.net [Electronic resource]. URL: https://techukraine.net/%D1%89%D0%BE-%D1%82%D0%B0%D0%BA%D0%B5-%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0-ipv4/ (accessed: 05.06.2023).
3. Що таке Ethernet - енциклопедія lanmarket.ua [Electronic resource]. URL: https://lanmarket.ua/ua/entsiklopediya/telekommunikatsionnye-tekhnologii/ethernet.html (accessed: 05.06.2023).
4. Що таке NAT? Для чого використовую даний стандарт? | Блог HyperHost.UA [Electronic resource]. URL: https://hyperhost.ua/info/uk/shcho-take-nat-dlya-chogo-vikoristovuyu-daniy-standart (accessed: 05.06.2023).
5. Все, що потрібно знати про IP-адресу [Electronic resource]. URL: https://maxnet.ua/blog/vse-chto-nuzhno-znat-ob-ip-adrese/ (accessed: 05.06.2023).
6. IPv4 і IPv6: розповідаємо про протоколи Інтернету — як працює система зсередини [Electronic resource]. URL: https://phoneinfo8.info/ipv4-i-ipv6-rozpovidaiemo-pro-protokoli-internety/ (accessed: 31.05.2023).
7. IPv6 security - Apple Support [Electronic resource]. URL: https://support.apple.com/en-euro/guide/security/seccb625dcd9/web (accessed: 05.06.2023).
8. АСТРА впроваджує IPv6 - Інтернет-провайдер Астра [Electronic resource]. URL: https://astra.in.ua/informatsiia/novyny/292-astra-vprovadzhuie-ipv6 (accessed: 05.06.2023).
9. Cisco Packet Tracer - Networking Simulation Tool [Electronic resource]. URL: https://www.netacad.com/courses/packet-tracer (accessed: 31.05.2023).

10. Використання Cisco Packet Tracer – UA5.org [Electronic resource]. URL: https://ua5.org/lan/1469-vykorystannya-cisco-packet-tracer.html (accessed: 05.06.2023).
11. Що таке C#? Чи підходить мені ця мова програмування? Чому вона крута? [Electronic resource]. URL: https://beetroot.academy/blog/courses/what-is-C (accessed: 31.05.2023).
12. Що таке локальна мережа? - блог інтернет-провайдера Briz в Одесі [Electronic resource]. URL: https://www.briz.ua/blog/article/cto-takoe-lokalnaya-set (accessed: 05.06.2023).
13. Що таке .NET і чим займаються .NET-розробники? | training.epam.ua [Electronic resource]. URL: https://training.epam.ua/News/Items/301?lang=ua (accessed: 31.05.2023).
14. Інформатика в школі - Програмування на C# [Electronic resource]. URL: https://informatics.in.ua/programming_csharp/part_01.php (accessed: 05.06.2023).
15. //visualbasic.at.ua/ ::: Програмування на Visual Basic ::: уроки по Visual Basic, практичні роботи по Visual Basic, презентації по Visual Basic, прогами Visual Basic, реферати та курсові роботи з програмування [Electronic resource]. URL: https://visualbasic.at.ua/index/teoretichnij_material/0-4 (accessed: 05.06.2023).
16. Режим емуляції в Cisco Packet Tracer 7.3.1 [Electronic resource]. URL: https://naurok.com.ua/rezhim-emulyaci-v-cisco-packet-tracer-7-3-1-237133.html (accessed: 31.05.2023).
17. WPF в порівнянні з WinForms - The complete WPF tutorial [Electronic resource]. URL: https://wpf-tutorial.com/uk/2/%D0%BF%D1%80%D0%BE-wpf/wpf-%D0%B2-%D0%BF%D0%BE%D1%80%D1%96%D0%B2%D0%BD%D1%8F%D0%BD%D0%BD%D1%96-%D0%B7-winforms/ (accessed: 05.06.2023).

ДОДАТОК А
ЛІСТИНГ КОДУ ВІДОБРАЖЕННЯ ТЕКСТОВИХ ПОЛЕЙ

```
private void UpdateAddressVisibility()
        {
            if (useYourAddressCheckBox.Checked)
            {
                int value = (int)amountOfRouters.Value;

                if (value == 2)
                {
                    ResetFormLittle();
                    Address1.Visible = true;
                    Address1Text.Visible = true;
                    Address2.Visible = true;
                    Address2Text.Visible = true;
                    Address3.Visible = true;
                    Address3Text.Visible = true;
                    this.Size = new Size(425, 286);
                }
                else if (value == 3)
                {
                    ResetFormLittle();
                    Address1.Visible = true;
                    Address1Text.Visible = true;
                    Address2.Visible = true;
                    Address2Text.Visible = true;
                    Address3.Visible = true;
                    Address3Text.Visible = true;
                    Address4.Visible = true;
                    Address4Text.Visible = true;
                    Address5.Visible = true;
                    Address5Text.Visible = true;
                    this.Size = new Size(425, 347);
                }
                else if (value == 4)
                {
                    ResetFormLittle();
                    Address1.Visible = true;
                    Address1Text.Visible = true;
                    Address2.Visible = true;
                    Address2Text.Visible = true;
                    Address3.Visible = true;
                    Address3Text.Visible = true;
                    Address4.Visible = true;
                    Address4Text.Visible = true;
                    Address5.Visible = true;
                    Address5Text.Visible = true;
                    Address6.Visible = true;
                    Address6Text.Visible = true;
                    Address7.Visible = true;
```

```
            Address7Text.Visible = true;
            this.Size = new Size(425, 411);
        }
        else if (value == 5)
        {
            ResetFormLittle();
            Address1.Visible = true;
            Address1Text.Visible = true;
            Address2.Visible = true;
            Address2Text.Visible = true;
            Address3.Visible = true;
            Address3Text.Visible = true;
            Address4.Visible = true;
            Address4Text.Visible = true;
            Address5.Visible = true;
            Address5Text.Visible = true;
            Address6.Visible = true;
            Address6Text.Visible = true;
            Address7.Visible = true;
            Address7Text.Visible = true;
            Address8.Visible = true;
            Address8Text.Visible = true;
            Address9.Visible = true;
            Address9Text.Visible = true;
            this.Size = new Size(425, 480);
        }
        else if (value == 6)
        {
            ResetFormLittle();
            Address1.Visible = true;
            Address1Text.Visible = true;
            Address2.Visible = true;
            Address2Text.Visible = true;
            Address3.Visible = true;
            Address3Text.Visible = true;
            Address4.Visible = true;
            Address4Text.Visible = true;
            Address5.Visible = true;
            Address5Text.Visible = true;
            Address6.Visible = true;
            Address6Text.Visible = true;
            Address7.Visible = true;
            Address7Text.Visible = true;
            Address8.Visible = true;
            Address8Text.Visible = true;
            Address9.Visible = true;
            Address9Text.Visible = true;
            Address10.Visible = true;
            Address10Text.Visible = true;
            Address11.Visible = true;
            Address11Text.Visible = true;
```

```
                this.Size = new Size(425, 550);
            }
            else
            {
                ResetFormLittle();
            }
        }
        else
        {
            ResetFormLittle();
        }
    }
```

ДОДАТОК Б
ЛІСТИНГ КОДУ ЩО ПОВЕРТАЄ ВІДОБРАЖЕННЯ ЗА ЗАМОВЧУВАННЯМ

```
private void ResetFormLittle()
        {
            Address1.Visible = false;
            Address1Text.Visible = false;
            Address2.Visible = false;
            Address2Text.Visible = false;
            Address3.Visible = false;
            Address3Text.Visible = false;
            Address4.Visible = false;
            Address4Text.Visible = false;
            Address5.Visible = false;
            Address5Text.Visible = false;
            Address6.Visible = false;
            Address6Text.Visible = false;
            Address7.Visible = false;
            Address7Text.Visible = false;
            Address8.Visible = false;
            Address8Text.Visible = false;
            Address9.Visible = false;
            Address9Text.Visible = false;
            Address10.Visible = false;
            Address10Text.Visible = false;
            Address11.Visible = false;
            Address11Text.Visible = false;
            this.Size = new Size(425, 195);
            generateButton.Size = new Size(351, 36);
        }
```

ДОДАТОК В
ЛІСТИНГ КОДУ КНОПКИ

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


            if (amountOfRouters.Value == 2)
            {
                if (useYourAddressCheckBox.Checked)
                {
                    string address1 = Address1.Text;
                    string address2 = Address2.Text;
                    string address3 = Address3.Text;

                    if (!string.IsNullOrWhiteSpace(address1) &&
                        !string.IsNullOrWhiteSpace(address2) &&
                        !string.IsNullOrWhiteSpace(address3))
                    {
                        string outputText =
TextForSetup2Routers(address1, address2, address3);

                        string filePath = "output.txt";
                        System.IO.File.WriteAllText(filePath,
outputText);

                        // Відкриття файлу

System.Diagnostics.Process.Start(filePath);

                        MessageBox.Show("Файл успішно створено та
відкрито.", "Успіх");
                    }
                    else
                    {
                        MessageBox.Show("Введіть значення в усі
текстові поля.", "Помилка");

                    }
                }
                else
                {
                    string address1 = GenerateIPv6Address();
                    string address2 = GenerateIPv6Address();
                    string address3 = GenerateIPv6Address();

                    string outputText =
TextForSetup2Routers(address1, address2, address3);

                    string filePath = "output.txt";
                    System.IO.File.WriteAllText(filePath,
```

```
outputText);

                    // Відкриття файлу
                    System.Diagnostics.Process.Start(filePath);

                    MessageBox.Show("Файл успішно створено та
відкрито.", "Успіх");

                }

            }
            else if (amountOfRouters.Value == 3)
            {
                if (useYourAddressCheckBox.Checked)
                {
                    string address1 = Address1.Text;
                    string address2 = Address2.Text;
                    string address3 = Address3.Text;
                    string address4 = Address4.Text;
                    string address5 = Address5.Text;

                    if (!string.IsNullOrWhiteSpace(address1) &&
                        !string.IsNullOrWhiteSpace(address2) &&
                        !string.IsNullOrWhiteSpace(address3) &&
                        !string.IsNullOrWhiteSpace(address4) &&
                        !string.IsNullOrWhiteSpace(address5))
                    {
                        string outputText =
TextForSetup3Routers(address1, address2, address3, address4,
address5);

                        string filePath = "output.txt";
                        System.IO.File.WriteAllText(filePath,
outputText);

                        // Відкриття файлу

System.Diagnostics.Process.Start(filePath);

                        MessageBox.Show("Файл успішно створено та
відкрито.", "Успіх");
                    }
                    else
                    {
                        MessageBox.Show("Введіть значення в усі
текстові поля.", "Помилка");

                    }
                }
                else
                {
```

```
                    string address1 = GenerateIPv6Address();
                    string address2 = GenerateIPv6Address();
                    string address3 = GenerateIPv6Address();
                    string address4 = GenerateIPv6Address();
                    string address5 = GenerateIPv6Address();

                    string outputText =
TextForSetup3Routers(address1, address2, address3, address4,
address5);

                    string filePath = "output.txt";
                    System.IO.File.WriteAllText(filePath,
outputText);

                    // Відкриття файлу
                    System.Diagnostics.Process.Start(filePath);

                    MessageBox.Show("Файл успішно створено та
відкрито.", "Успіх");

                }

            }
            else if (amountOfRouters.Value == 4)
            {
                if (useYourAddressCheckBox.Checked)
                {
                    string address1 = Address1.Text;
                    string address2 = Address2.Text;
                    string address3 = Address3.Text;
                    string address4 = Address4.Text;
                    string address5 = Address5.Text;
                    string address6 = Address6.Text;
                    string address7 = Address7.Text;

                    if (!string.IsNullOrWhiteSpace(address1) &&
                        !string.IsNullOrWhiteSpace(address2) &&
                        !string.IsNullOrWhiteSpace(address3) &&
                        !string.IsNullOrWhiteSpace(address4) &&
                        !string.IsNullOrWhiteSpace(address5) &&
                        !string.IsNullOrWhiteSpace(address6) &&
                        !string.IsNullOrWhiteSpace(address7))
                    {
                        string outputText =
TextForSetup4Routers(address1, address2, address3, address4,
address5, address6, address7);

                        string filePath = "output.txt";
                        System.IO.File.WriteAllText(filePath,
outputText);
```

```
                        // Відкриття файлу
System.Diagnostics.Process.Start(filePath);

                        MessageBox.Show("Файл успішно створено та
відкрито.", "Успіх");
                    }
                    else
                    {
                        MessageBox.Show("Введіть значення в усі
текстові поля.", "Помилка");

                    }
                }
                else
                {
                    string address1 = GenerateIPv6Address();
                    string address2 = GenerateIPv6Address();
                    string address3 = GenerateIPv6Address();
                    string address4 = GenerateIPv6Address();
                    string address5 = GenerateIPv6Address();
                    string address6 = GenerateIPv6Address();
                    string address7 = GenerateIPv6Address();

                    string outputText =
TextForSetup4Routers(address1, address2, address3, address4,
address5, address6, address7);

                    string filePath = "output.txt";
                    System.IO.File.WriteAllText(filePath,
outputText);

                    // Відкриття файлу
                    System.Diagnostics.Process.Start(filePath);

                    MessageBox.Show("Файл успішно створено та
відкрито.", "Успіх");

                }

            }
            else if (amountOfRouters.Value == 5)
            {
                if (useYourAddressCheckBox.Checked)
                {
                    string address1 = Address1.Text;
                    string address2 = Address2.Text;
                    string address3 = Address3.Text;
                    string address4 = Address4.Text;
                    string address5 = Address5.Text;
                    string address6 = Address6.Text;
```

```
                    string address7 = Address7.Text;
                    string address8 = Address8.Text;
                    string address9 = Address9.Text;

                    if (!string.IsNullOrWhiteSpace(address1) &&
                        !string.IsNullOrWhiteSpace(address2) &&
                        !string.IsNullOrWhiteSpace(address3) &&
                        !string.IsNullOrWhiteSpace(address4) &&
                        !string.IsNullOrWhiteSpace(address5) &&
                        !string.IsNullOrWhiteSpace(address6) &&
                        !string.IsNullOrWhiteSpace(address7) &&
                        !string.IsNullOrWhiteSpace(address8) &&
                        !string.IsNullOrWhiteSpace(address9))
                    {
                        string outputText =
TextForSetup5Routers(address1, address2, address3, address4,
address5, address6, address7, address8, address9);

                        string filePath = "output.txt";
                        System.IO.File.WriteAllText(filePath,
outputText);

                        // Відкриття файлу

System.Diagnostics.Process.Start(filePath);

                        MessageBox.Show("Файл успішно створено та
відкрито.", "Успіх");
                    }
                    else
                    {
                        MessageBox.Show("Введіть значення в усі
текстові поля.", "Помилка");

                    }
                }
                else
                {
                    string address1 = GenerateIPv6Address();
                    string address2 = GenerateIPv6Address();
                    string address3 = GenerateIPv6Address();
                    string address4 = GenerateIPv6Address();
                    string address5 = GenerateIPv6Address();
                    string address6 = GenerateIPv6Address();
                    string address7 = GenerateIPv6Address();
                    string address8 = GenerateIPv6Address();
                    string address9 = GenerateIPv6Address();

                    string outputText =
TextForSetup5Routers(address1, address2, address3, address4,
address5, address6, address7, address8, address9);
```

```
                    string filePath = "output.txt";
                    System.IO.File.WriteAllText(filePath,
outputText);

                    // Відкриття файлу
                    System.Diagnostics.Process.Start(filePath);

                    MessageBox.Show("Файл успішно створено та
відкрито.", "Успіх");

                }

            }
            else if (amountOfRouters.Value == 6)
            {
                    if (useYourAddressCheckBox.Checked)
                    {
                        string address1 = Address1.Text;
                        string address2 = Address2.Text;
                        string address3 = Address3.Text;
                        string address4 = Address4.Text;
                        string address5 = Address5.Text;
                        string address6 = Address6.Text;
                        string address7 = Address7.Text;
                        string address8 = Address8.Text;
                        string address9 = Address9.Text;
                        string address10 = Address10.Text;
                        string address11 = Address11.Text;

                        if (!string.IsNullOrWhiteSpace(address1)
&&
                            !string.IsNullOrWhiteSpace(address2)
&&
                            !string.IsNullOrWhiteSpace(address3)
&&
                            !string.IsNullOrWhiteSpace(address4)
&&
                            !string.IsNullOrWhiteSpace(address5)
&&
                            !string.IsNullOrWhiteSpace(address6)
&&
                            !string.IsNullOrWhiteSpace(address7)
&&
                            !string.IsNullOrWhiteSpace(address8)
&&
                            !string.IsNullOrWhiteSpace(address9)
&&

!string.IsNullOrWhiteSpace(address10)&&
                            !string.IsNullOrWhiteSpace(address11))
```

```
                        {
                            string outputText =
TextForSetup6Routers(address1, address2, address3, address4,
address5, address6, address7, address8, address9, address10,
address11);

                            string filePath = "output.txt";
                            System.IO.File.WriteAllText(filePath,
outputText);

                            // Відкриття файлу

System.Diagnostics.Process.Start(filePath);

                            MessageBox.Show("Файл успішно створено
та відкрито.", "Успіх");
                        }
                        else
                        {
                            MessageBox.Show("Введіть значення в
усі текстові поля.", "Помилка");

                        }
                    }
                    else
                    {
                        string address1 = GenerateIPv6Address();
                        string address2 = GenerateIPv6Address();
                        string address3 = GenerateIPv6Address();
                        string address4 = GenerateIPv6Address();
                        string address5 = GenerateIPv6Address();
                        string address6 = GenerateIPv6Address();
                        string address7 = GenerateIPv6Address();
                        string address8 = GenerateIPv6Address();
                        string address9 = GenerateIPv6Address();
                        string address10 = GenerateIPv6Address();
                        string address11 = GenerateIPv6Address();

                        string outputText =
TextForSetup6Routers(address1, address2, address3, address4,
address5, address6, address7, address8, address9, address10,
address11);

                        string filePath = "output.txt";
                        System.IO.File.WriteAllText(filePath,
outputText);

                        // Відкриття файлу

System.Diagnostics.Process.Start(filePath);
```

```
                        MessageBox.Show("Файл успішно створено та
відкрито.", "Успіх");

                    }

            }
            else
            {
                MessageBox.Show("Не виконані умови для виконання
генерації", "Поммлка");
            }
        }
```