МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: «Програмний додаток проектування шляху прокладання трубопроводу природнього газу»

Здобувачки групи ІТ-01 Недайхліб Марії Сергіївни
(шифр групи) (прізвище, ім'я, по батькові)

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

_____ Марія НЕДАЙХЛІБ
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник В.о. завідувача кафедри, к.т.н., доцент, Світлана ВАЩЕНКО _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«__» _____ 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Недайхліб Марії Сергіївні

1 Тема роботи *Програмний додаток проектування шляху прокладання трубопроводу природнього газу*

керівник роботи *Ващенко Світлана Михайлівна, к.т.н., доцент*,

затверджені наказом по університету від «*07*» травня 2024 р. №0482-VI

2 Строк подання студентом роботи « 26 » травня 2024 р.

3 Вхідні дані до роботи технічне завдання на розробку проекту

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області (огляд останніх досліджень та публікаці, аналіз програмних продуктів-аналогів, постановка задачі), моделювання та проектування (моделювання, проектування інформаційної системи, проектування моделі бази даних), програмна реалізація (архітектура програмного продукту, програмна реалізація, використання програмного продукту, тестування), висновки

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність досліджень, об'єкт та предмет досліджень, мета та постановка задачі, аналіз аналогів програмних продуктів, порівняльна таблиця програм-аналогів, функціональні вимоги, діаграма варіантів використання, структурно-функціональний аналіз, контекстна діаграма в нотації IDEF0, діаграма декомпозиції, засоби реалізації, практична реалізація, тестування програмного додатку, висновки

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання 24.07.2023

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підготовка специфікації. Аналіз предметної області. Розробка ТЗ. Моделювання додатку	24.07.2023 – 03.10.2023	
2	Моделювання додатку: структурно-функціональні діаграми, діаграма варіантів використання, діаграма класів	04.10.2023 – 21.11.2023	
3	Проектування та реалізація бази даних	22.11.2023 – 08.01.2024	
4	Розробка програмного додатку визначення оптимального шляху прокладанян гадопроводу: шаблону додатку, модулю введення інформації, модулю обчислень	09.01.2024 – 13.05.2024	
5	Тестування програмного додатку та виправлення помилок	14.05.2024 – 21.05.2024	
6	Підготовка пояснювальної записки	22.05.2024 – 26.05.2024	

Студент _____ Марія НЕДАЙХЛІБ
(підпис)

Керівник роботи _____ к.т.н., доц. Світлана ВАЩЕНКО
(підпис)

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Програмний додаток проектування шляху прокладання трубопроводу природнього газу».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 31 найменування, додатків. Загальний обсяг роботи – 125 сторінки, у тому числі 58 сторінок основного тексту, 3 сторінки списку використаних джерел, 66 сторінок додатків.

Актуальність роботи полягає в відсутності ефективних та зручних інструментів проектування шляху прокладання трубопроводу, згідно аналізу та розрахунку термодинамічних параметрів газового потоку в трубі. З урахуванням практичного застосування отриманих даних, додаток відкриває нові перспективи для вдосконалення технічних рішень та забезпечення оптимальної ефективності газотранспортних систем та інших інженерних проектів

Мета роботи: розробка програмного додатку аналізу та розрахунку термодинамічних процесів для проектування шляху прокладання трубопроводу природнього газу.

Моделювання та проектування додатку включало в себе створення відповідної архітектури та моделі бази даних для забезпечення ефективного зберігання і управління інформацією. У процесі програмної реалізації було використано класи, а також додано новий клас DataBase для взаємодії з базою даних MySQL через інструмент Workbench.

Додаток успішно реалізовано і має два основних режими роботи: "Ввести інформацію" та "Переглянути інформацію". Розроблений програмний додаток є ефективним інструментом для вирішення задач з обчислення оптимального шляху прокладання трубопроводу і розрахунку термодинамічних параметрів газового потоку в трубі, та відображення результатів користувачу.

Ключові слова: шлях прокладання, швидкість газу в трубі, термодинаміка, потік в трубопроводі.

ANNOTATION

The topic of the bachelor's thesis is “Software application for designing the route of a natural gas pipeline”.

The explanatory note consists of an introduction, 3 chapters, conclusions, a list of 31 references, and appendices. The total volume of the work is 125 pages, including 58 pages of the main text, 3 pages of the list of references, and 66 pages of appendices.

The relevance of the work lies in the lack of effective and convenient tools for designing the pipeline route, according to the analysis and calculation of the thermodynamic parameters of the gas flow in the pipe. Taking into account the practical application of the data obtained, the application opens up new prospects for improving technical solutions and ensuring the optimal efficiency of gas transportation systems and other engineering projects.

The analysis of analogous software products made it possible to identify their advantages and limitations, which will help to improve and determine the unique characteristics of our software for analyzing the thermodynamics of gas flow in a pipe. The main criteria of the created application are an easy interface, no dependence on the Internet and direct focus of the developed product on the gas industry.

Objective: to develop a software application for analyzing and calculating thermodynamic processes for designing a natural gas pipeline route.

In order to calculate the thermodynamic parameters of gas flow in a pipe, input data such as pipe cross-sectional area, pipe radius, temperature, initial pressure, and other parameters are used.

To calculate the optimal pipeline route, the transportation problem solving method is used. The calculation process is based on the Bellman-Ford algorithm and involves determining the optimal pipeline route to transport a certain volume of natural gas from the starting point (supply) to the end point (withdrawal) through a network of pipelines with different capacity and radii, using the minimum amount of time. The input data are the number of points between which pipeline sections can be laid, the capacity and radius of each connection between 2 points, and the total volume of gas to be delivered.

Designing the main scenarios of user interaction with the developed software product, namely various possibilities for data entry, obtaining results, and other key functions aimed at meeting user needs, which allowed us to design key classes, their attributes, methods, and relationships between them. They were further used to get a complete overview of the program structure during the development process, determine the ways of interaction between its components, and establish the general context for further programming and development.

The database of the developed software product contains the calculated results of the task of finding the optimal pipeline route. Also, thermodynamic processes were calculated for each selected pipe section. In the process of program implementation, classes were used, and a new DataBase class was added to interact with the MySQL database through the Workbench tool.

The software application is designed according to the client-database model, where all computing and information logic is concentrated in the client part, which directly interacts with the database, allowing for efficient data maintenance and storage, providing convenient and quick access to the calculation results through the client interface.

For the software implementation of the client side, we used the C++ program language in the Visual Studio 2022 environment and the additional component CppWinFormsProjectTemplate. Initially, an interface was developed that allows the user to conveniently and efficiently enter input data for further processing and use in the software application. Next, the programmatic implementation of all the predefined required application classes was carried out.

The implementation of the testing stage is aimed at ensuring its quality and compliance with the requirements. This process reflected that the entered data in the developed software application is subjected to verification for correctness, and additional confirmation was obtained from external sources that all calculations are performed correctly.

The application has been successfully implemented and has two main modes of operation: “Enter information” and “View information”. The developed software application is an effective tool for solving problems of calculating the optimal pipeline

route and calculating the thermodynamic parameters of gas flow in the pipe, and displaying the results to the user.

The application of modern methods and innovative approaches in further research promises even greater progress in this direction, and the program can become a key tool for scientists and engineers in studying and solving challenges related to the thermodynamics of gas flows in pipes.

Keywords: optimal laying path, gas velocity in the pipe, thermodynamics, flow in the pipeline.

ЗМІСТ

ВСТУП

Сучасний період визначається стрімким розвитком технологій та постійним прогресом у наукових дослідженнях, що породжує потребу вдосконалення та оптимізації різноманітних процесів. Однією з вирішальних галузей, що відіграє визначальну роль у цьому контексті, є термодинаміка газових потоків. З метою досягнення оптимальних результатів та впровадження ефективних рішень у різних галузях, починаючи від промисловості й закінчуючи науковими дослідженнями, важливим стає розуміння та вивчення термодинамічних процесів, що відбуваються в газових трубопроводах.

Незважаючи на важливість цього аспекту, виявлення відсутності ефективних та зручних інструментів для аналізу та розрахунку термодинамічних параметрів газового потоку в трубі залишається актуальною проблемою. У зв'язку з цим, на передній план виступає необхідність розробки нового програмного забезпечення, призначеного для детального аналізу та ефективного розрахунку термодинамічних процесів в газових потоках з метою проектування шляху прокладання трубопроводу.

Об’єкт дослідження – розрахункове забезпечення проектування шляху прокладання трубопроводу природнього газу.

Предмет дослідження – технології розроблення програмних додатків проектування шляху прокладання трубопроводу природнього газу.

Мета даної роботи – створення додатку як інструменту, який дозволить інженерам, науковцям та фахівцям в області термодинаміки газів здійснювати точні та ефективні розрахунки різних термодинамічних параметрів газових потоків в трубі з метою проектування оптимального шляху прокладання трубопроводу. Основні завдання додатку включають в себе розрахунок масового потоку, швидкості газу, тиску, температури, адіабатичної ентальпії та інших ключових характеристик.

Для досягнення мети проекту необхідно виконати наступні задачі:

– дослідити предметну область та провести аналіз існуючих аналогів онлайн сервісів для визначення пріоритетних пунктів при розрахунку термодинамічних процесів у розробленому додатку;

– провести моделювання робочих процесів щодо виконання розрахунків. змоделювати майбутній програмний продукт;

– виконати проєктування бази даних для збереження необхідної інформації;

– розробити модулі введення вхідних даних для розрахунку, модулів обчислення термодинамічних процесів та модуль виведення результатів;

– реалізувати зв’язок вхідних даних (результатів розрахунків) з базою даних.

Розроблене програмне забезпечення має на меті спростити та прискорити аналіз термодинамічних процесів газового потоку в трубі, забезпечуючи точні та достовірні результати. З урахуванням практичного застосування отриманих даних, додаток відкриває нові перспективи для вдосконалення технічних рішень та забезпечення оптимальної ефективності газотранспортних систем та інших інженерних проектів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1Огляд останніх досліджень і публікацій

Останні роки характеризуються інтенсивним розвитком досліджень у галузі термодинаміки газових потоків та їх впливу на різноманітні технологічні та інженерні процеси [1]. Велика увага приділяється розробці програмних інструментів для точного моделювання та аналізу цих процесів.

Однією з перспективних галузей є розробка програм, які дозволяють ефективно вирішувати завдання, пов'язані з термодинамічними параметрами газових потоків в трубах. Останні дослідження вказують на значущість розуміння та оптимізації цих параметрів для покращення ефективності газотранспортних систем, енергетичних установок та інших інженерних об'єктів [2].

Важливим внеском у цю галузь є розробка нового програмного забезпечення, спрямованого на детальний аналіз та ефективний розрахунок термодинамічних процесів в газових потоках. Цей напрямок досліджень відкриває нові можливості для вирішення практичних завдань у різних сферах, починаючи від промисловості та закінчуючи науковими дослідженнями [3].

Окрім того, публікації доповідей та проведення конференцій [1], впровадження цілеспрямованих профільних заходів (такі як конгреси) [2] свідчать про активний інтерес до розробок у галузі термодинамічного аналізу газових потоків. Застосування сучасних методів та інноваційних підходів у подальших дослідженнях обіцяє ще більший прогрес у цьому напрямку, а програма може стати ключовим інструментом для вчених та інженерів у вивченні та вирішенні викликів, пов'язаних із термодинамікою газових потоків в трубах.

Літературні джерела, що були вивчені [1-5], глибоко розглядають аспекти термодинаміки газового потоку в трубі та їх практичні застосування. Зокрема, "Термодинаміка газового потоку в трубі" [4] представляє електронну книгу, яка не

лише визначає основні закони та рівняння, але й надає приклади розв'язання задач з різних областей техніки, таких як авіація, ракетна техніка та теплоенергетика.

Методичний посібник для практичних занять з курсу "Термодинаміка газового потоку" [5] детально розглядає теоретичний матеріал щодо газового потоку та термодинамічних процесів. Містить практичні завдання, пов'язані з термодинамікою газового потоку для поглиблення знань та надання можливостей виконувати дослідження даних процесів.

Ключові слова в роботі:

Діабатичний процес - це термодинамічний процес, який відбувається без обміну теплом з оточуючим середовищем. В такому процесі ентропія системи залишається константною.

Діабатичні процеси можуть бути адіабатичними розширеннями або стисканнями газів без обміну теплом з оточуючим середовищем.

Ізохорний процес - це термодинамічний процес, в якому об'єм системи залишається незмінним. У такому процесі відбувається обмін теплом, але без виконання роботи.

Ізентропічний процес - це ідеалізований термодинамічний процес, при якому ентропія системи залишається постійною. Такий процес найбільше наближений до адіабатичного процесу, але може включати обмін теплом.

Температура при адіабатичному процесі - це температура, яка визначається при адіабатичному розширенні або стисканні газу. Залежно від напрямку процесу, ця температура може змінюватися.

Швидкість газу в трубі - це швидкість, з якою газ рухається вздовж трубопроводу. Вона може бути визначена як відносна або абсолютна, залежно від врахування швидкості самого газу та швидкості відносно оточуючого середовища.

1.2Аналіз програмних продуктів-аналогів

Вивчення програмних аналогів є ключовим етапом у визначенні ринкових рішень та їх відповідності вимогам та цілям нашого програмного продукту для аналізу термодинаміки газового потоку в трубі. В рамках цього дослідження ми ретельно проаналізували наступні програмні рішення, які мають аналогічні функціональні можливості.

1. OpenFOAM [6]:

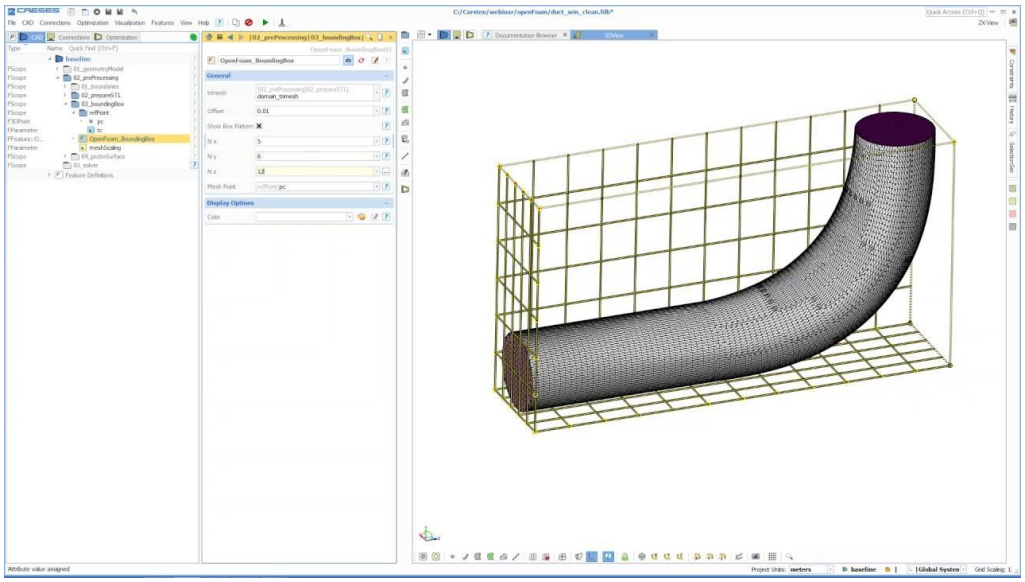


Рисунок 1.1 – Програма OpenFOAM

Опис:

OpenFOAM – це відкрите програмне забезпечення для чисельного моделювання газових та рідких потоків з інструментами для аналізу термодинамічних процесів та гідродинаміки.

Переваги:

Відкритий код та безкоштовний доступ.

Розширені можливості чисельного моделювання.

Недоліки:

Може бути складним для початківців.

2. COMSOL Multiphysics [7]:

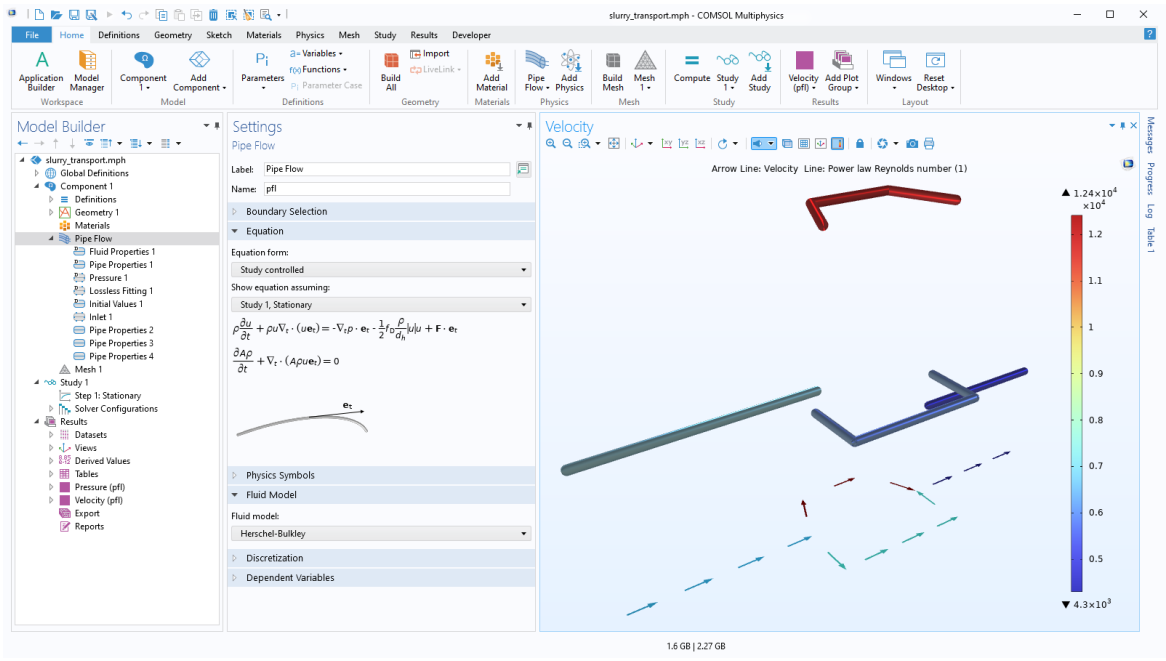


Рисунок 1.2 – Програма COMSOL Multiphysics

Опис:

COMSOL Multiphysics - це програмний продукт для чисельного моделювання теплових та гідравлічних явищ, використовуваний для аналізу газових потоків.

Переваги:

Можливість моделювання різних фізичних явищ.

Недоліки:

Комерційна ліцензія.

3. ANSYS Fluent [8]:

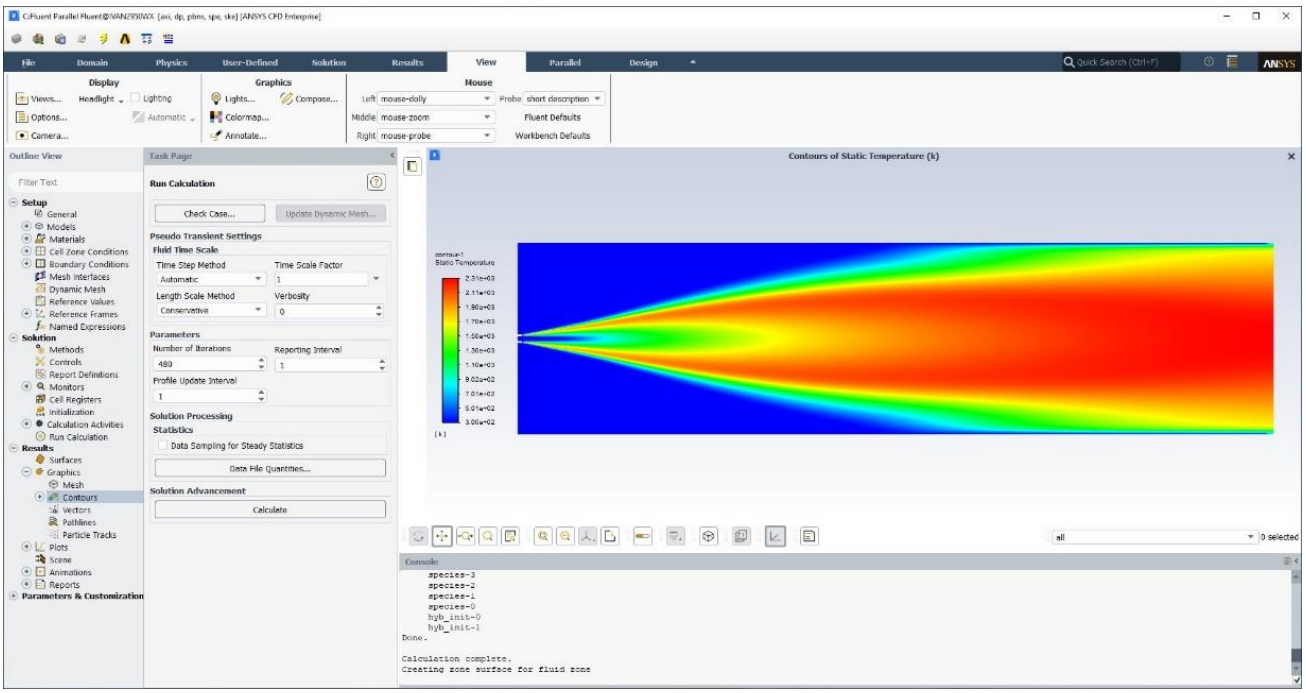


Рисунок 1.3 – Програма ANSYS

Опис:

ANSYS Fluent - програмний пакет для чисельного моделювання теплообміну та газових потоків.

Переваги:

Висока точність та розширені можливості моделювання.

Інтеграція з іншими інженерними інструментами ANSYS.

Недоліки:

Великі обчислювальні витрати.

Комерційна ліцензія.

4. SimScale [9]:

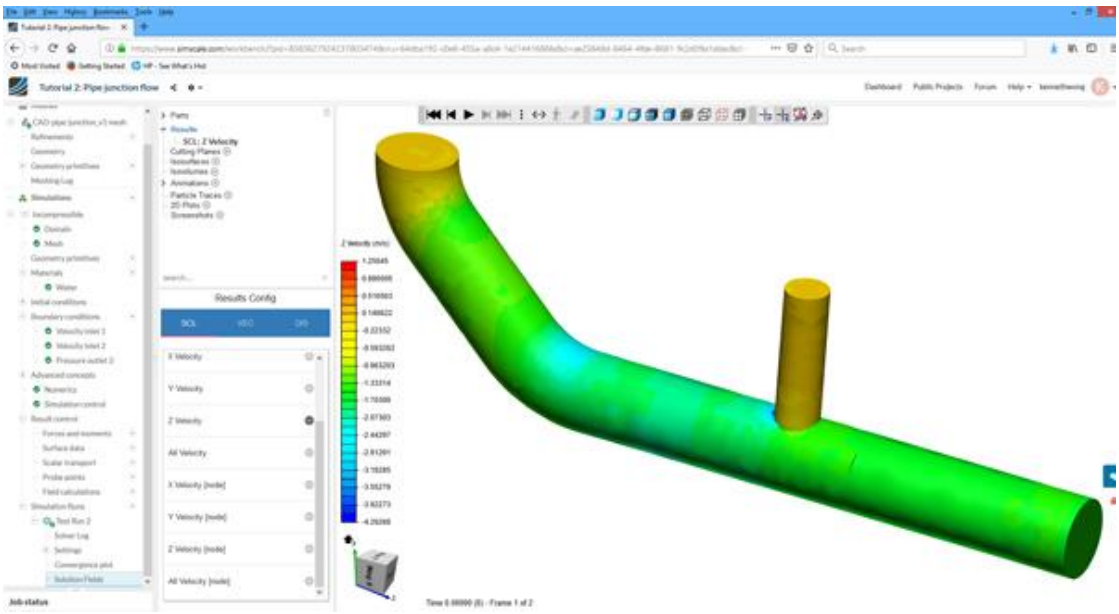


Рисунок 1.4 – Програма SimScale

Опис:

SimScale - це хмарний сервіс для чисельного моделювання, який включає в себе різноманітні можливості для аналізу термодинамічних явищ.

Переваги:

Доступ через Інтернет та спрощений інтерфейс.

Можливість спільної роботи в реальному часі.

Недоліки:

Обмежені можливості у безкоштовній версії.

Залежність від інтернет-підключення.

Таблиця 1.1 – Порівняльна таблиця додатків-аналогів

Назва додатку	**Критерії оцінювання**		
	Доступний інтерфейс	**Залежність від інтернету**	**Зосередженість на газовій сфері**
OpenFOAM	+	-	-
COMSOL Multiphysics	-	-	+
ANSYS Fluent	-	-	+
SimScale	-	+	-

Аналіз програмних продуктів-аналогів надав можливість виявити їх переваги та обмеження, що допоможе у вдосконаленні та визначенні унікальних характеристик нашого програмного забезпечення для аналізу термодинаміки газового потоку в трубі. Основними критеріями створеного додатку є легкий інтерфейс, відсутність залежності від інтернету та безпосередня зосередженість розробленого продукту на газовій сфері.

1.3Постановка задачі

Мета розробки програмного забезпечення полягає в створенні інструменту аналізу та розрахунку термодинамічних процесів для проектування шляху прокладання трубопроводу.

Для проведення розрахунків термодинамічних параметрів газового потоку в трубі використовуються вхідні дані, такі як густина газу, площа поперечного перерізу труби, радіус труби, температура, початковий тиск, та інші параметри.

Розробити обчислення масового потоку, швидкості газу, тиску, температури, адіабатичної ентальпії та інших важливих термодинамічних характеристик на основі введених користувачем параметрів. Визначені необхідні формули [6]:

– масовий потік газу:

$$mass\_flow = \rho \cdot V \cdot \pi \cdot r^2 \quad (1)$$

де mass_flow – масовий потік газу (витрата газу, що проходить через поперечний переріз труби в одиницю часу), кг/с;

ρ – густина газу, кг/м^3;

V – швидкість потоку, м/с;

π, r – константне число Пі та радіус поперечного перерізу труби, для обчислення площі поперечного перерізу труби, м.

– швидкість газу:

$$velocity = \frac{0.1}{\rho A} \quad (2)$$

де velocity – швидкість газового потоку;

ρ – густина газу, кг/м^3;

A – площа поперечного перерізу, м^3.

– формула Бернулі, для розрахунку тиску газу:

$$pressure = \frac{1}{2} \cdot \rho \cdot V^2 \cdot \pi \cdot r^2 \quad (3)$$

де pressure – тиск газу, розрахований за формулою Бернулі, Па;

ρ, V, π, r – використані величини згідно формули (1).

– обчислення температури газу:

$$temperature = \frac{pressure}{\rho \cdot C_p} \quad (4)$$

де temperature – температура потоку газу, К;

pressure – тиск газу, розрахований за формулою (3);

ρ – густина газу, кг/м^3;

C_p – питома теплоємність газу, Дж/(кг*К).

– адіабатична ентальпія (водночас ізентропічна ефективність обчислюється за допомогою відношення адіабатичної ентальпії до ізентропічної ентальпії):

$$adiabatic\_enthalpy = C_p \cdot T \cdot \left(\frac{pressure}{pressure\_initial}\right) \quad (5)$$

де adiabatic_enthalpy – адіабатична ентальпія газу, Дж/кг;
pressure – тиск газу, розрахований за формулою (3);
pressure_initial – початковий (вхідний) тиск газу, Па;
T – температура газу розрахована за формулою (4);
C_p – питома теплоємність газу, Дж/(кг*К).

– ізохорний тиск та знаходиться за наступною формулою:

$$isochoric\_pressure = pressure\_initial \cdot \left(\frac{temperature}{temperature\_initial}\right) \quad (6)$$

де $isochoric\_pressure$ – тиск потоку газу при ізохорному процесі, Па;
pressure_initial – початковий (вхідний) тиск газу, Па;
temperature – температура газу розрахована за формулою (4);
temperature_initial – початкова (вхідна) температура газу, К.

– адіабатична температура розраховується як:

$$adiabatic\_temperature = temperature\_initial \cdot \left(\frac{pressure}{pressure\_initial}\right) \quad (7)$$

де adiabatic_temperature – температура газу при адіабатичному процесі, К;
pressure – тиск газу, розрахований за формулою (3);
pressure_initial – початковий (вхідний) тиск газу, Па;
temperature – температура газу розрахована за формулою (4);
temperature_initial – початкова (вхідна) температура газу, К.

Метою розробки є створити інструмент, який задовольнить потреби інженерів, науковців та фахівців у галузі термодинаміки газів, спростить аналіз газових потоків та забезпечить точні та ефективні розрахунки.

Для обчислення оптимального шляху прокладання трубопроводу використовується спосіб вирішення транспортної задачі. Процес розрахунку побудований за алгоритмом Беллмана-Форда [31].

Задача полягає у визначенні оптимального шляху прокладання трубопроводу для транспортування певного об’єму природнього газу від початкової точки (подачі) до кінцевої (відбору) через мережу трубопроводів з різною пропускною спроможністю та радіусами, використавши мінімальну кількість часу.

При розрахунку оптимального шляху використовуються наступні етапи:

1. Внесення вхідних даних:
    - кількість точок;
    - пропускна здатність та радіус кожного з’єднання між 2 точками;
    - загальний об’єм газу, який необхідно доставити.
2. Ініціалізація:
    - Відстань до початкової точки встановлюється 0, до всіх інших точок вона визначена як нескінченність
3. Проходження системи:
    - Переглядаються всі точки та враховуються всі відстані для кожного їх з’єднання, результуючий аналіз запам’ятовується;
    - Виконується повторна перевірка ділянок трубопроводів (з’єднання між певними точками) на можливість зменшення відстані.
4. Визначення оптимального шляху:
    - Обирається оптимальне досягання необхідного рівня транспортованого об’єму газу за допомогою певних частин трубопроводів, згідно заданих користувачем початкових умов.

Алгоритм Беллмана-Форда (етапи 2-3) забезпечує точне рішення задачі знаходження найкоротшого шляху в графі, навіть у випадках з негативними вагами

ребер. Алгоритм гарантує, що всі відстані та попередники будуть обчислені правильно після проходження всіх ребер графу n – 1 разів, де n — кількість вузлів у графі. Це забезпечує високу точність у визначенні найкоротших шляхів.

Повне технічне завдання на розробку міститься в додатку А.

2 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

Моделювання - це метод, що передбачає створення абстракційного відображення реального об'єкта чи системи для аналізу, розуміння та прогнозування їхньої поведінки [12]. Цей інструментарій використовується для відтворення ключових аспектів об'єкта та експериментування з ними, щоб мати змогу передбачати майбутній розвиток об’єкта чи системи, та оптимізувати процеси.

2.1 Моделювання в IDEF0

Задача структурно-функціонального моделювання полягає у визначенні та формалізації взаємодії між різними компонентами системи.

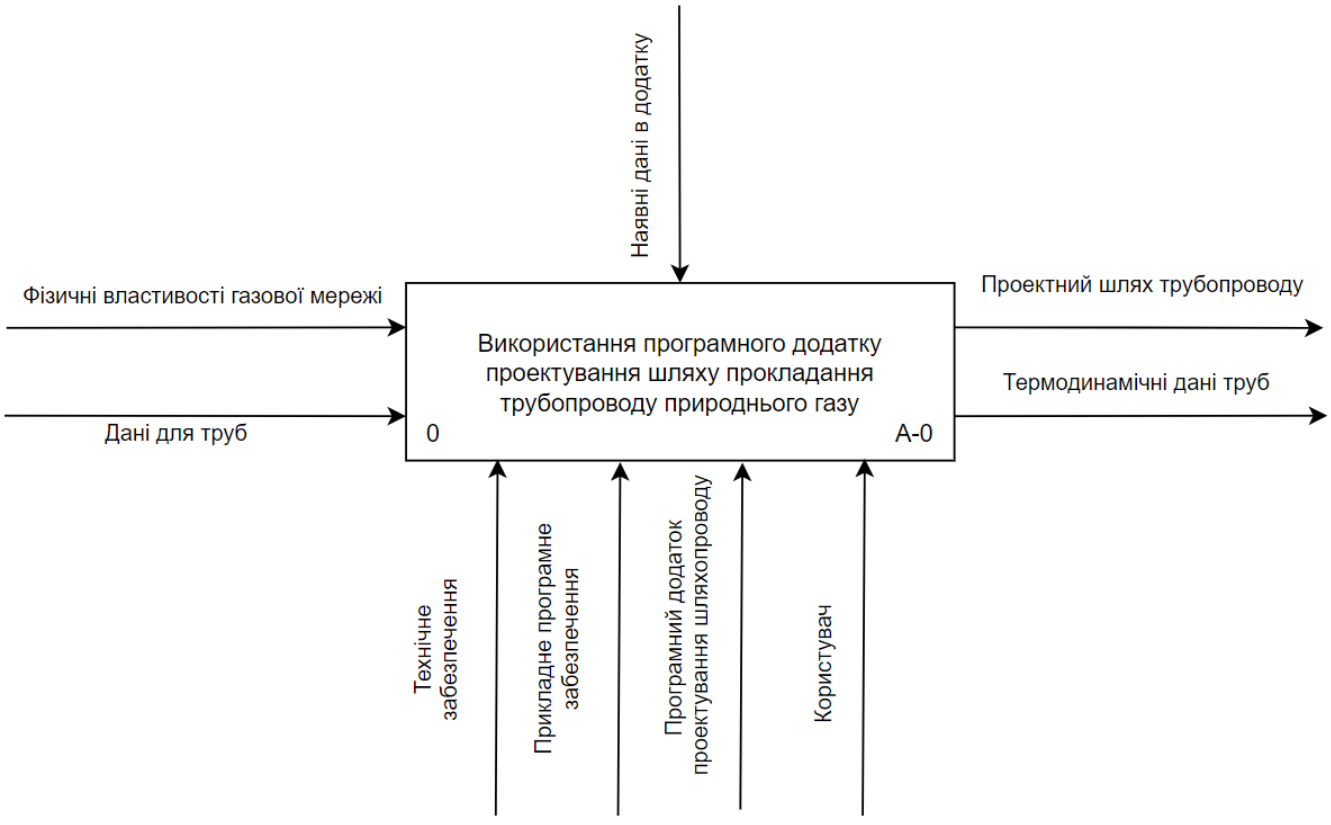


Рисунок 2.1 – Контекстна діаграма

Для переходу від контекстної діаграми до діаграми декомпозиції, розбиваємо основну функцію системи на підфункції, які виконують конкретні завдання. Контекстна діаграма відображає загальну функцію системи, а діаграма декомпозиції деталізує її на більш дрібні компоненти для кращого розуміння процесів і їх взаємодії.

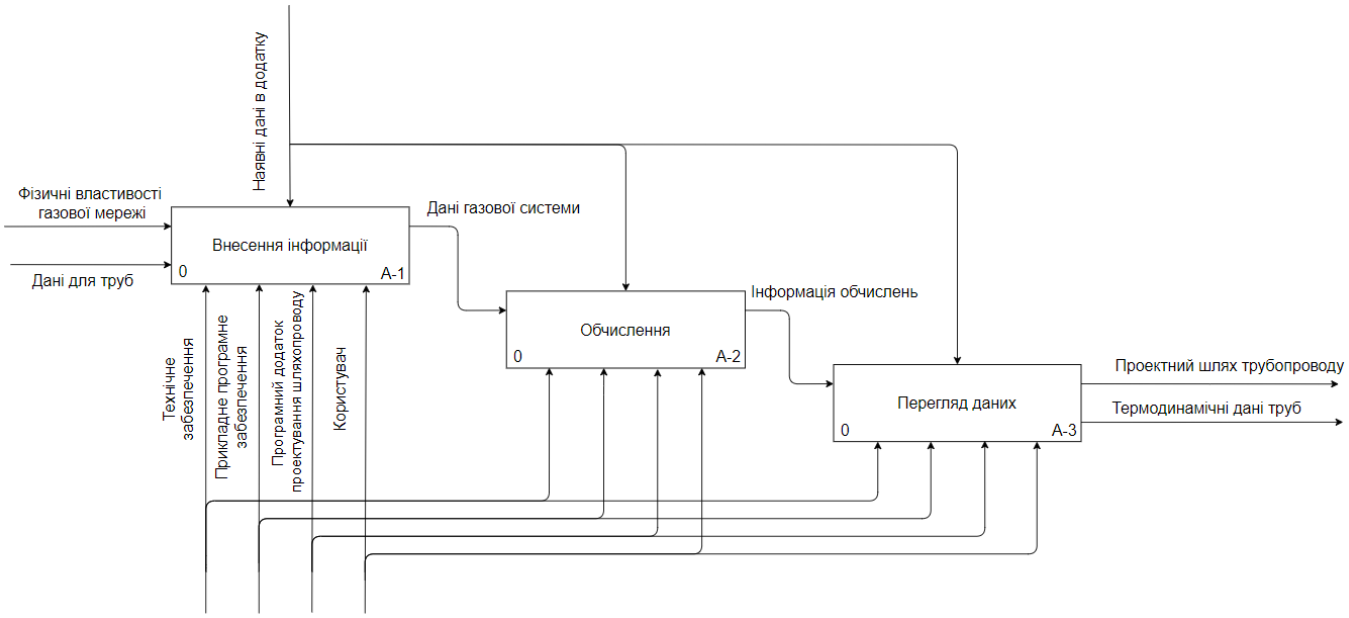


Рисунок 2.2 – Діаграма декомпозиції

Переходячи до діаграми декомпозиції другого рівня, ми ще детальніше розподіляємо кожну з підфункцій, наприклад, функцію "Обчислення" на "Розрахунок термодинаміки" та "Оптимізація шляху". Це необхідно для глибшого аналізу та оптимізації окремих складових процесу, що допомагає визначити можливі місця покращення і забезпечити ефективніше управління проектом.

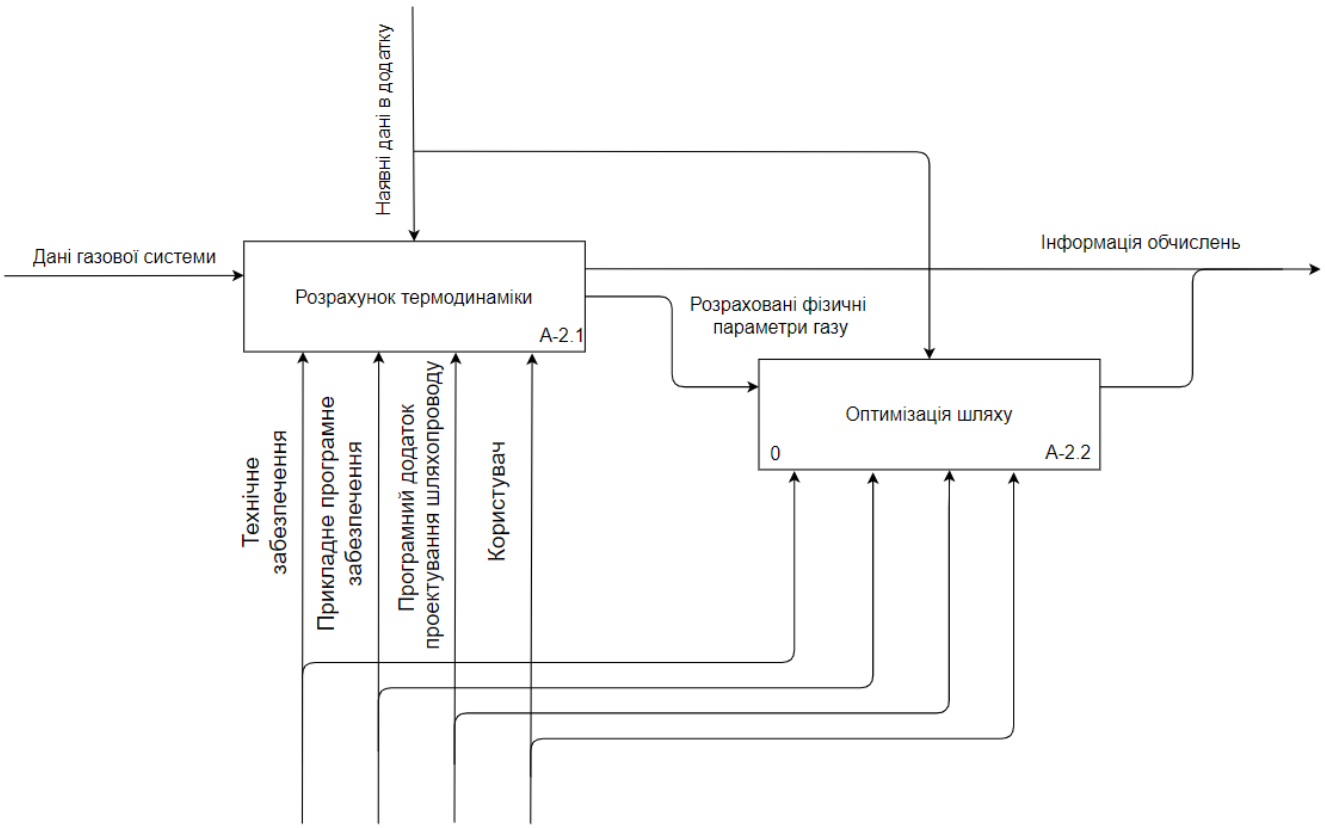


Рисунок 2.3 – Діаграма декомпозиції другого рівня процесу обчислення

Перехід від контекстної діаграми до діаграми декомпозиції дозволяє систематично та детально аналізувати ключові функції системи. Таке розбиття на підфункції допомагає глибше зрозуміти та проаналізувати кожен компонент, що є важливим для успішної розробки та впровадження програмного забезпечення. Декомпозиція на детальніші рівні допомагає виявити потенційні проблеми на ранніх етапах, оптимізувати роботу системи та забезпечити її ефективну роботу. Це сприяє досягненню основних цілей проекту та підвищує якість кінцевого продукту.

2.2 Діаграма варіантів використання

Діаграми варіантів використання створюються для того, щоб забезпечити високорівневе уявлення про функціональні можливості системи та показати, як різні актори (користувачі або зовнішні системи) взаємодіють з цією системою. Вони

допомагають визначити вимоги до системи, виявити можливі взаємодії та полегшити комунікацію між учасниками проекту [13].

Далі буде зображена таблиця варіантів використання (табл. 2.1), яка відобразить основні сценарії взаємодії користувача з розробленим програмним продуктом. Це включатиме різні можливості введення даних, отримання результатів, та інші ключові функції, спрямовані на задоволення потреб користувача в аналізі термодинамічних процесів газового потоку в трубі.

Таблиця 2.1 – Варіанти використання системи

Код	**Основний актор**	**Найменування**	**Формування**
К1	Користувач	Переглянути сторінку з внесенням даних	Користувач може переглядати сторінку, на якій вноситься інформація для обрахунку термодинаміки газового потоку в трубі та дані для обчислення оптимізаційної задачі.
К2	Користувач	Переглянути сторінку з результатами	Користувач може переглядати сторінку, на якій виводиться інформація обрахованої термодинаміки
К3	Користувач	Внести інформацію	Учасник може вводити потрібну інформацію як цілим числом так і дробовим.
К4	Користувач	Обчислити	Користувач може обчислити термодинаміку за внесеними даними.
К5	Користувач	Додати запис до бази даних	Користувач може додавати при потребі обраховані дані в базу даних
К6	Користувач	Переглянути базу даних	Користувач може переглянути усі дані, які він записав в базу даних.
К7	Користувач	Розрахунок термодинаміки	Користувач може переглянути термодинамічні дані розраховані на введеній та наявній в додатку інформації.

Продовження таблиці 2.1

К8	Користувач	Оптимізація шляху	Користувач може переглянути оптимальний шлях прокладання трубопроводу розрахований з вхідних даних.

Згідно з табличною формою варіантів використання, для кращого візуального сприйняття зв’язків між актором та варіантами використанням можна створити діаграму варіантів використання. Актори та варіанти використання, які зображені у вигляді діаграми, можна побачити той функціонал, який необхідно реалізувати в системі.

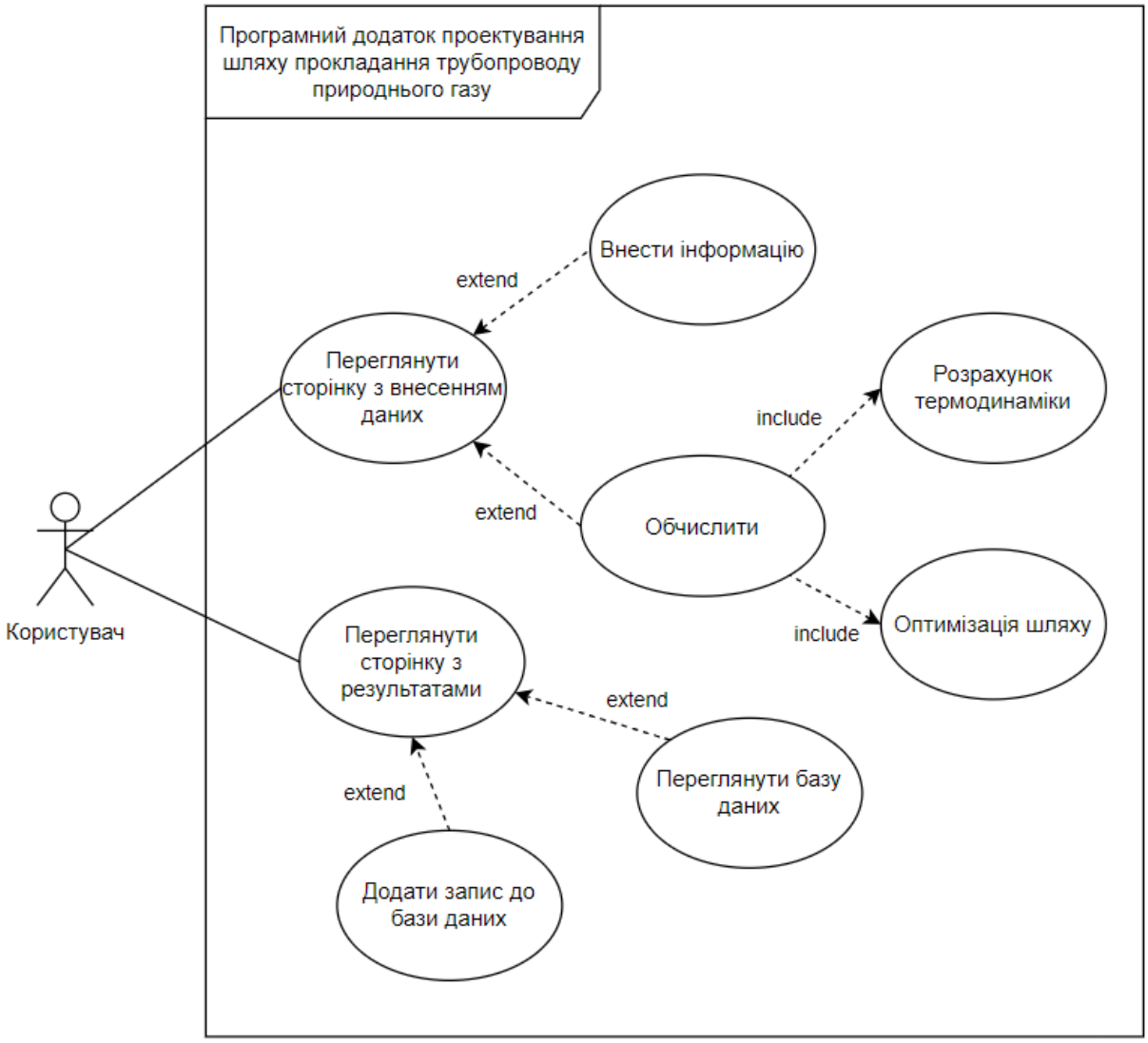


Рисунок 2.4 – Діаграма варіантів використання

Після виявлення варіантів використання можна приступити до формування сутностей:

Сутність MyForm – призначена для виведення головного меню та зберігає в собі інофмацію для обчислення та обчислену інформацію.

Сутність Input – призначенна для приймання інформації та повертає її головній сутності MyForm.

Сутність Output – призначена для приймання обрахованої інформації та виводить її на екран.

Сутність Thermodynamics – призначена для обраховування термодинаміки газового потоку в трубі. Також зберігає введені та обраховані дані.

Сутність DBOutput – призначена для виведення в заданій області інформації з бази даних.

Сутність DataBase – призначена для роботи з базою даних.

2.3 Проектування бази даних

Класи є необхідними для майбутньої програмної реалізації, оскільки вони дозволяють структурувати код у логічні блоки, що полегшує його підтримку та розширення. Вони також сприяють повторному використанню коду і забезпечують інкапсуляцію даних, що підвищує безпеку і стабільність програмного забезпечення.

Для роботи з базою даних планується виконати розробку класів.

Клас DBOutput – призначений для виведення в задану область усіх даних з бази даних, які були збережені користувачем.

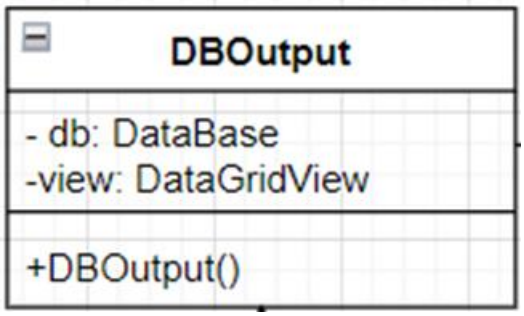


Рисунок 2.5 – Клас DBOutput

Клас DataBase – призначений для з’єднання з базою даних, добавленням записів в базу даних та отриманням інформації з бази даних.

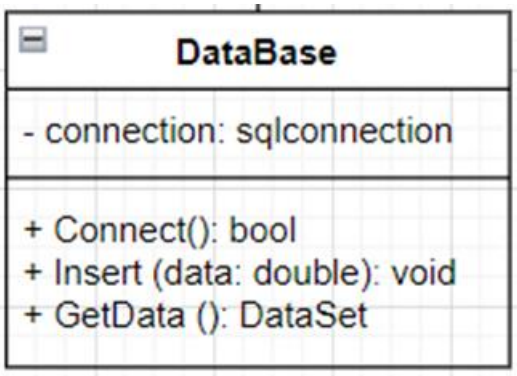


Рисунок 2.6 – Клас DataBase

Після опису класів, які будуть використовуватись в базі даних, ми можемо зобразити діаграму класів:

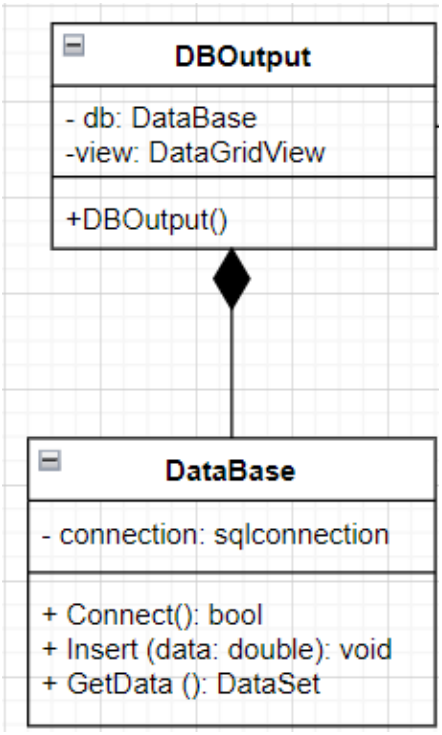


Рисунок 2.7 – Діаграма класів для бази даних

2.4 Проектування інформаційної системи

З усією інформацією про систему, отриманою у минулому підрозділі, можна розпочати розробку інформаційної системи, визначаючи також класи, якими вона буде володіти. Після ідентифікації всіх класів системи, буде побудовано діаграму класів, використовуючи мову UML.

Клас MyForm (рис. 2.8) відповідає за виведення головного вікна та області введення і виведення інформації (режими роботи програми). Включає в себе клас Thermodynamics.

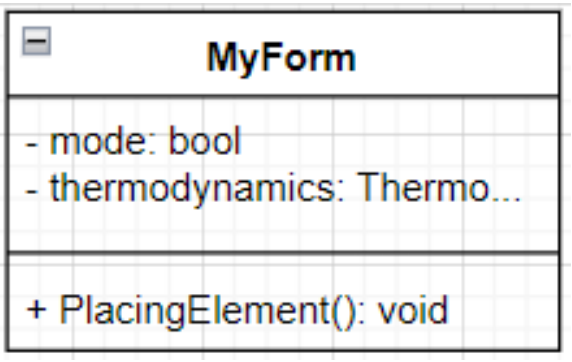


Рисунок 2.8 – Клас MyForm

Клас Input (рис. 2.9) відповідає за приймання тексту від користувача та має функцію для повернення цієї інформації головному класу MyForm.

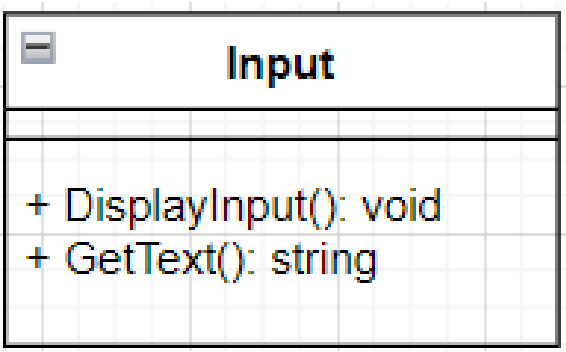


Рисунок 2.9 – Клас Input

Клас Output (рис. 2.10) відповідає за приймання інформації з головного класу MyForm в параметрах методу та виводить задані дані в область обрахованої термодинаміки для газового потоку в трубі.

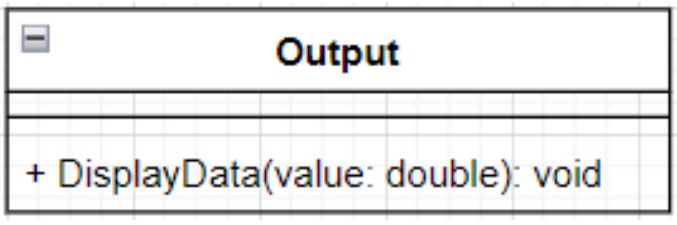


Рисунок 2.10 – Клас Output

Клас Thermodynamics (рис. 2.11) відповідає за обрахування термодинаміки та реалізованого алгоритму пошуку оптимального шляху прокладання трубопроводу на основі даних від користувача.

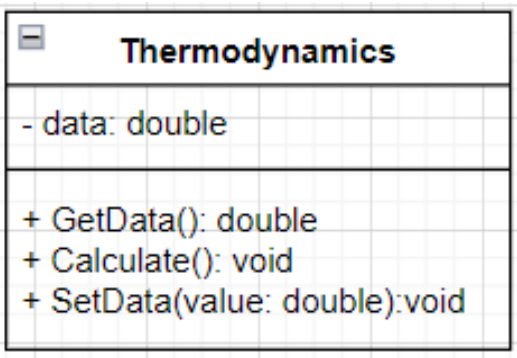


Рисунок 2.11 – Клас Thermodynamics

Описавши всі класи, можемо побудувати діаграму класів, до якої можна звертатись в подальшій розробці (рис. 2.12).

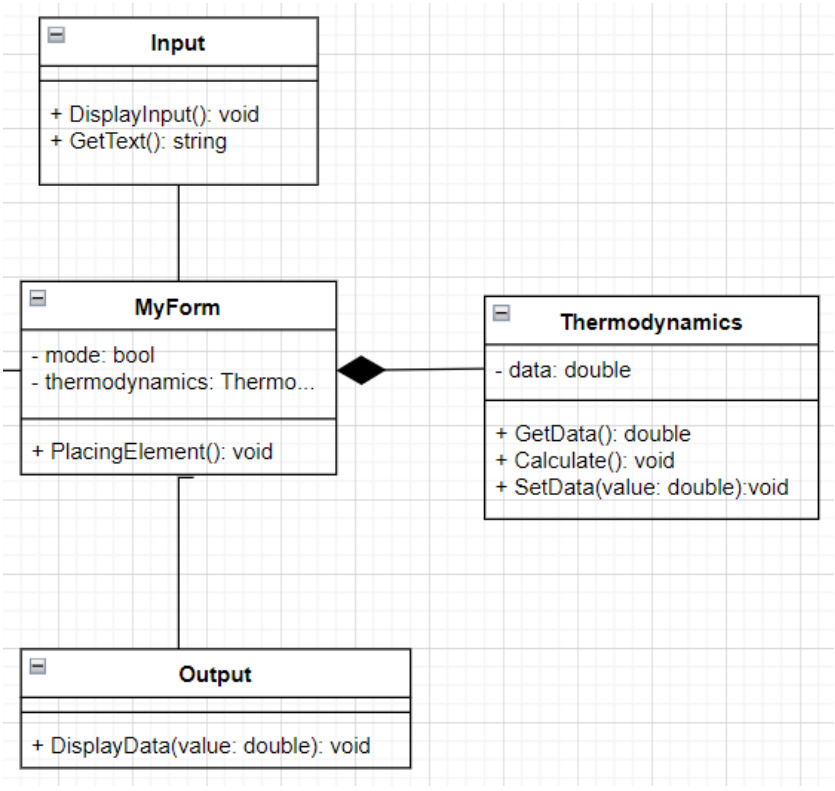


Рисунок 2.12 – Діаграма класів для програми обрахунку газового потоку в трубі

Цілісна діаграма класів - це графічне візуальне представлення структури класів та їх взаємодій у межах системи чи програмного продукту. Вона відображає ключові класи, їх атрибути, методи та зв'язки між ними. Цілісна діаграма класів допомагає в процесі розробки отримати повний огляд структури програми, визначити способи взаємодії між її складовими частинами та встановити загальний контекст для подальшого програмування та розробки.

Після визначення всіх класів та зв'язків для забезпечення працездатності програмної реалізації та роботи з бази даних, ми можемо створити цілісну діаграму класів для системи обрахунку газового потоку в трубі.

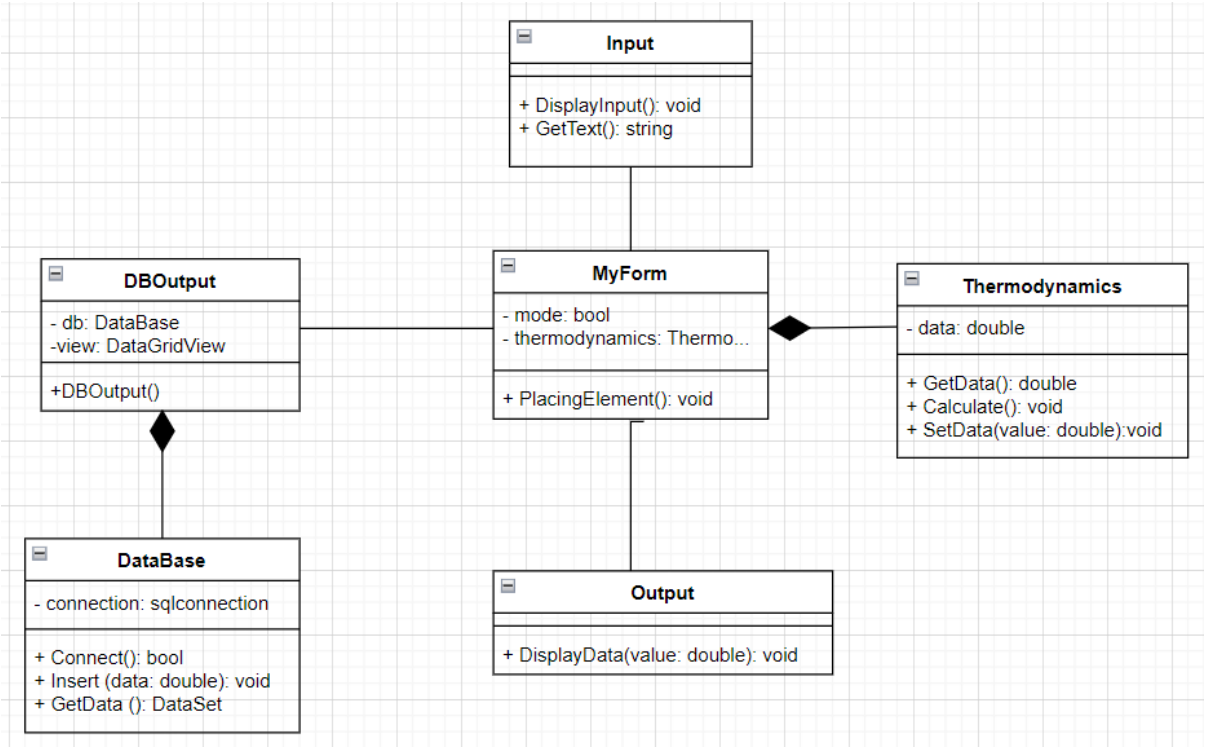


Рисунок 2.13 – Цілісна діаграма класів програмної реалізації

2.5 База даних

База даних розробленого програмного продукту вміщує в собі обраховані результати задачі пошуку оптимального шляху прокладання трубопроводу. Також, виконані розрахунки термодинамічних процесів для кожної обраної ділянки труби.

Схема бази даних відображає створені 2 таблиці у відповідності до розрахованих даних: таблиця термодинамічних властивостей трубопроводу та таблиці вихідних даних по обраному оптимізаційному шляху прокладання. Перша таблиця містить дані порядкового номеру ділянки труби, масового потоку, тиску, температури, адіабатичної ентальпії, ізохоричного тиску, адіабатичної температури. Друга, оптимальна, таблиця містить дані порядкового номеру ділянки труби, її пропускної здатності, радіусу, площі поперечного перерізу.

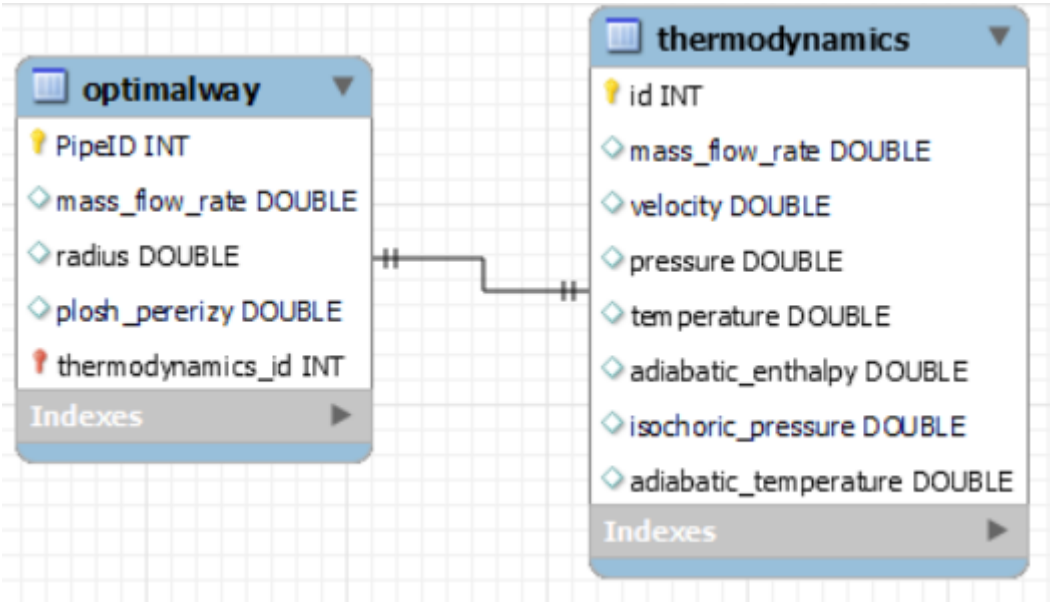


Рисунок 2.14 – Схема бази даних

2.6 Архітектура програмного додатку

Програмний додаток спроектовано за моделлю "клієнт-база даних", де вся обчислювальна та інформаційна логіка зосереджена в клієнтській частині, яка безпосередньо взаємодіє з базою даних [14].

Розглянемо складові цієї архітектури.

1. Клієнтська частина.
    - Обчислення термодинамічних процесів: Клієнтська частина є центром обчислень, де виконуються всі необхідні розрахунки, пов'язані з термодинамікою газового потоку в трубі. Вона надає інтерактивний інтерфейс для введення вхідних даних, таких як густини газу, радіус труби, початковий тиск, тощо, і виводу результатів розрахунків, таких як масовий потік, температура, тиск.
    - Взаємодія з базою даних: Клієнт напряму взаємодіє з базою даних для зберігання обчислених параметрів та отримання збережених даних. Це включає в себе функціонал для запису нових результатів обчислень в базу даних та читання існуючих записів для подальшого використання.
    - Перегляд даних: Крім того, користувач має можливість переглядати та аналізувати раніше збережені результати розрахунків. Це робиться через інтерфейс клієнта, де можна відобразити та здійснити докладний аналіз термодинамічних параметрів, збережених у базі даних.
2. База даних.
    - Зберігання та отримання даних: База даних використовується для ефективного зберігання усіх результатів обчислень та термодинамічних параметрів. Вона забезпечує можливість не лише зберігання даних, але й їхнього швидкого та точного відтворення при необхідності.

- Оптимізація доступу: База даних розроблена таким чином, щоб оптимізувати доступ до даних та спростити процес взаємодії клієнта з базою даних.

Ця архітектурна модель дозволяє ефективно обслуговувати та зберігати дані, забезпечуючи зручний та швидкий доступ до результатів обчислень через інтерфейс клієнта.

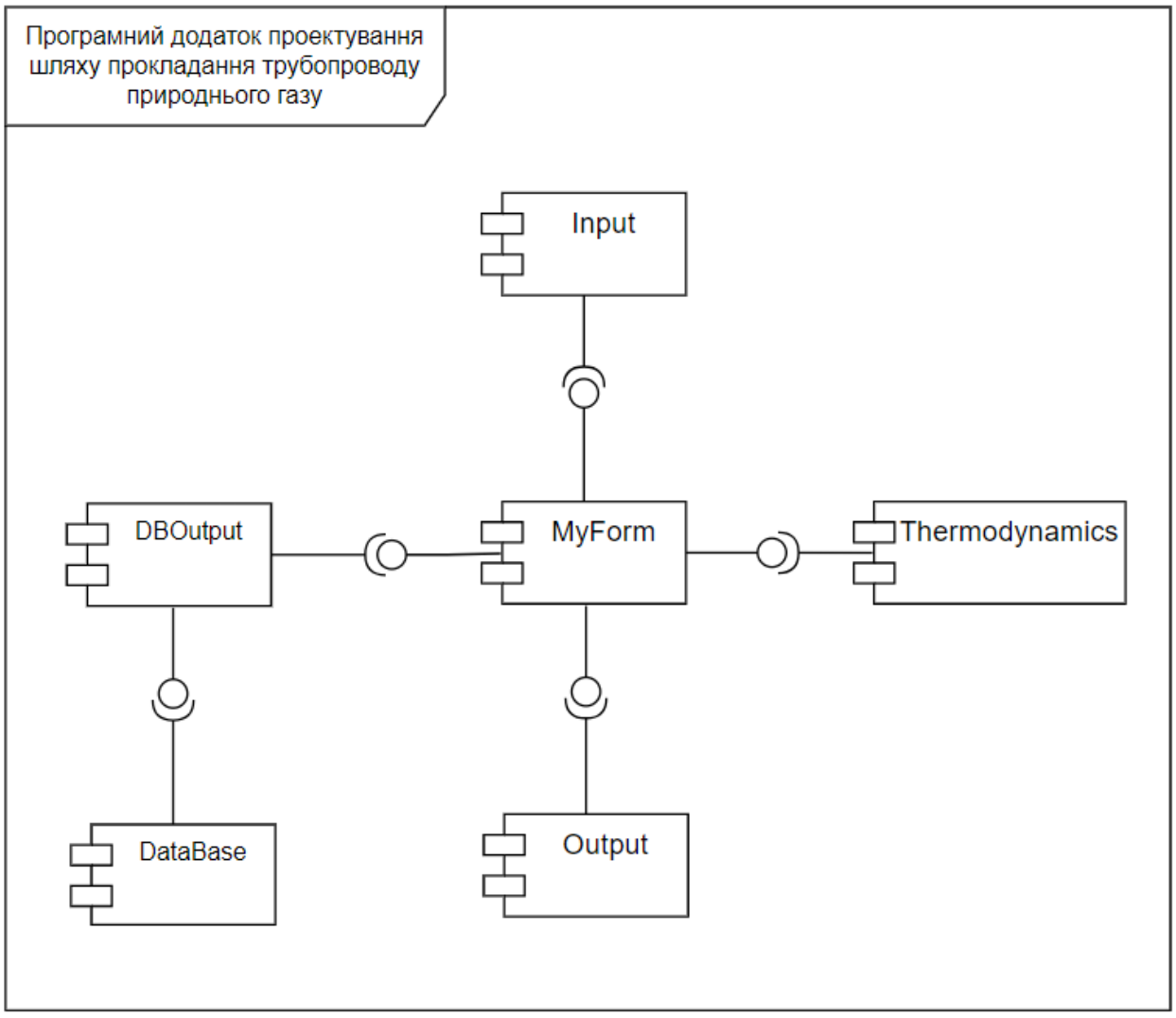


Рисунок 2.15 – Архітектура модулів системи

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Реалізація бази даних

MySQL обрано для створення бази даних з ряду причин, зокрема через його надійність, високу ефективність та розширені можливості. Загальні відомості про MySQL включають його статус як системи управління реляційними базами даних (СУБД) і відкритість та безкоштовність.

Основні риси MySQL, які враховані при виборі [13]:

- надійність: MySQL відзначається високою стабільністю та надійністю, особливо завдяки використанню транзакційного підходу, що забезпечує консистентність даних;
- ефективність: MySQL визнаний своєю продуктивністю та швидкодією обробкою запитів. Він оптимізований для роботи з великими обсягами даних та високою конкурентоспроможністю;
- відкритість та безкоштовність: Як відкрите програмне забезпечення, MySQL доступний для безкоштовного використання і має відкритий вихідний код, що дає можливість користувачам адаптувати його під свої потреби;

Для зручного взаємодії з базою даних MySQL у розробці програмного додатку було обрано MySQL Workbench — інтуїтивно зрозумілий графічний інструмент для управління базами даних. Його використання спростить взаємодію з базою даних у контексті розробки програмного додатку, забезпечуючи зручність та ефективність у процесі роботи.

Першим етапом в роботі з базою даних у MySQL Workbench є створення схеми бази даних. Схема визначає логічну структуру бази даних, включаючи таблиці, їхні поля та взаємозв'язки. Створення схеми є критичним етапом, оскільки вона визначає основні параметри для подальшої роботи з даними.

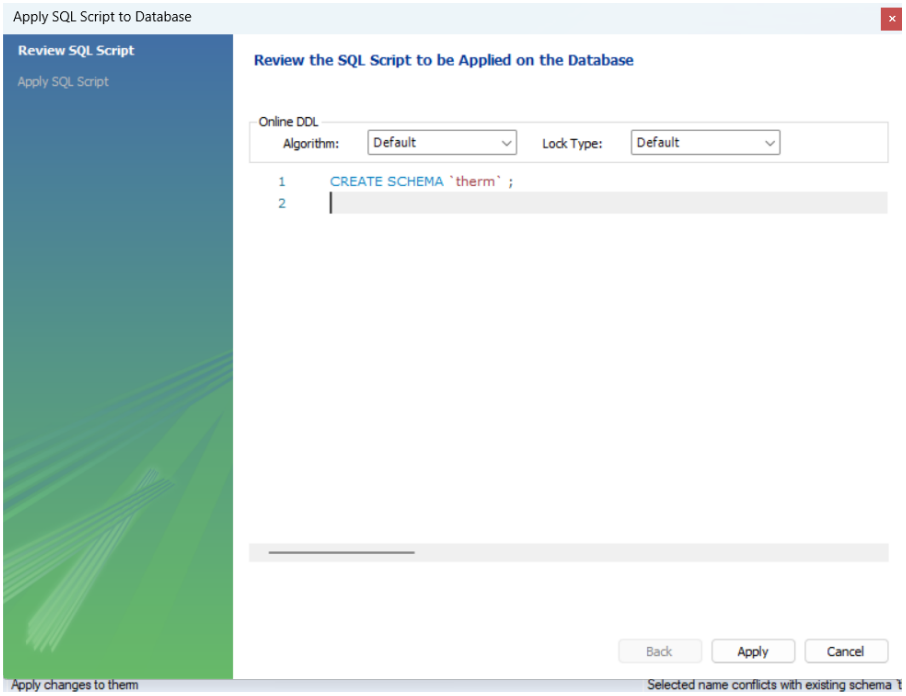


Рисунок 3.1 – Створення схеми для бази даних

Після створення схеми бази даних у MySQL Workbench за допомогою SQL-запиту можна переконатися, що створення відбулося успішно.

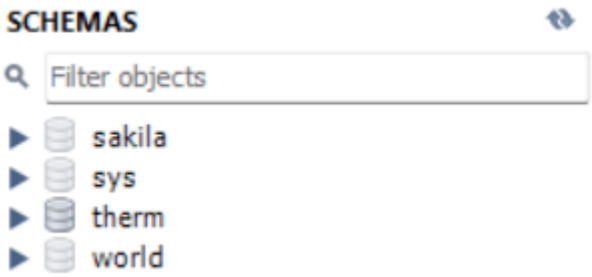


Рисунок 3.2 – Створена схема для бази даних

Перш ніж розпочати наступні кроки у процесі роботи з базою даних, необхідно визначити, з якою саме схемою будуть проводитись операції.

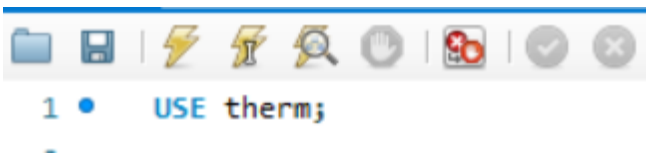


Рисунок 3.3 – Вибір схеми

Другим етапом у процесі створення бази даних є створення таблиць для визначеної схеми. Кожна таблиця представляє собою структурований набір даних, організованих у вигляді рядків та стовпців. Визначення полів (стовпців) та їхніх типів, а також визначення відносин між таблицями визначає структуру бази даних.

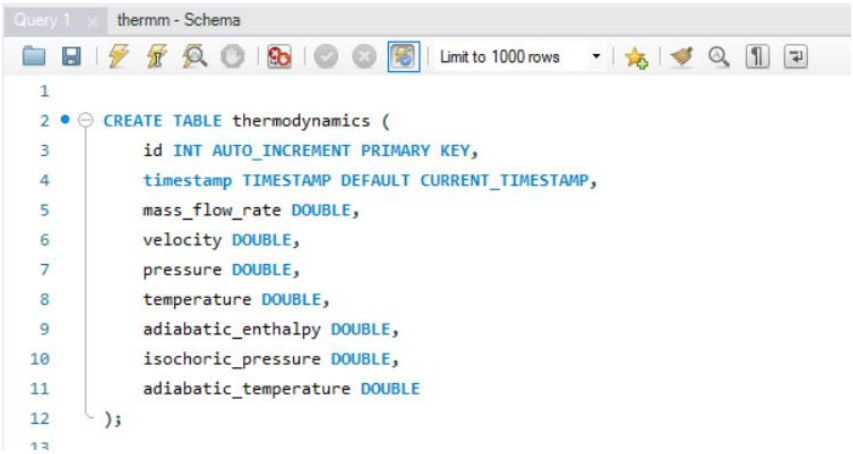


Рисунок 3.4 – Створення таблиці

Наступним кроком є заповнення бази даних необхідною інформацією та виведення цих даних на екран.

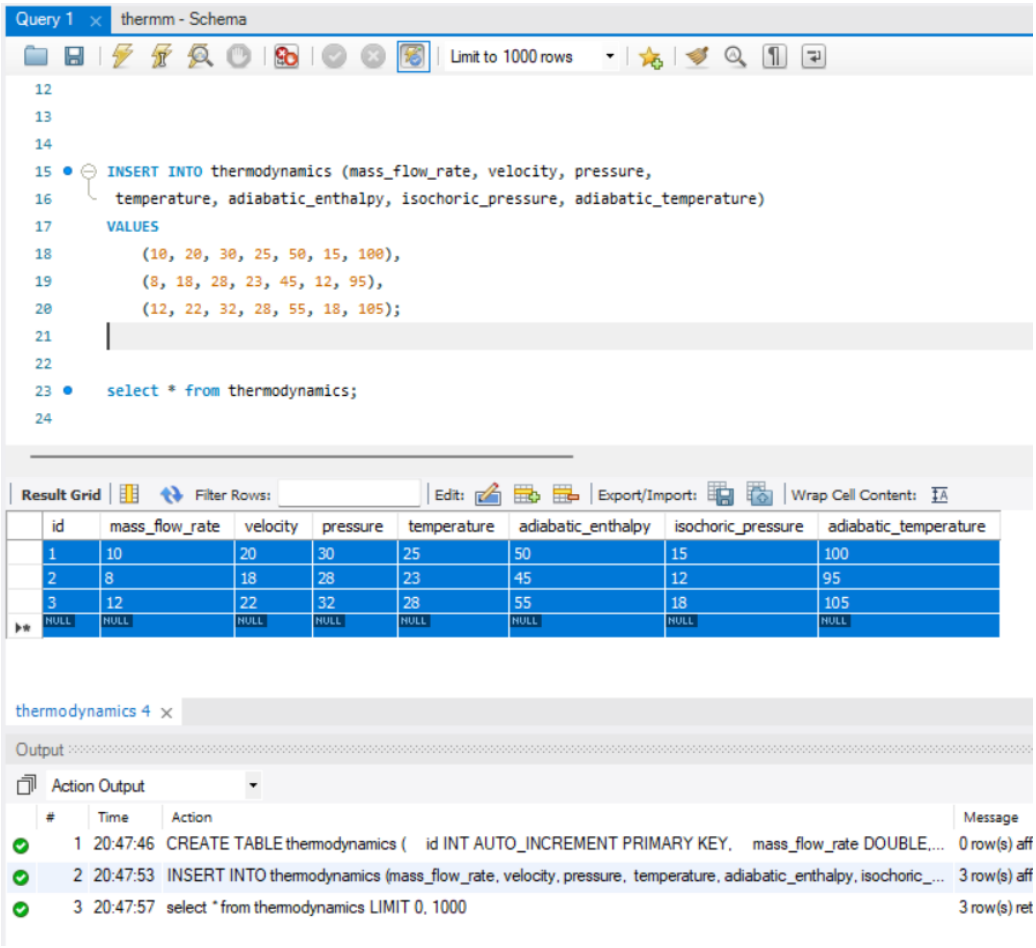


Рисунок 3.5 – Заповнення та перегляд даних в таблиці

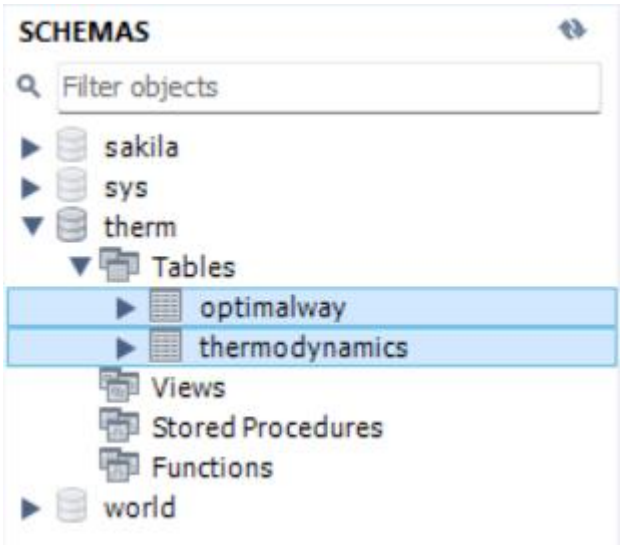


Рисунок 3.6 – Перелік таблиць бази даних

3.2 Програмна реалізація

3.2.1 Налаштування робочого середовища

Для створення програмного додатку було обрано Visual Studio 2022. Це інтегроване середовище розробки (IDE), розроблене Microsoft, яке надає розширені інструменти та можливості для розробки різноманітних програмних проектів. Деякі ключові аспекти Visual Studio 2022:

- можливість встановлення різних розширень та плагінів для адаптації IDE під конкретні потреби розробника;
- Visual Studio 2022 має ряд корисних інструментів для розробки, таких як редактор коду, дебагер, дизайнер інтерфейсу, система контролю версій, аналізатори продуктивності, інструменти тестування та багато інших;
- використання Windows Forms для швидкої та зручної розробки графічного інтерфейсу для програм на мові C++;
- Visual Studio регулярно оновлюється, щоб розробники отримували доступ до нових можливостей та вдосконалень, що відображають останні технологічні тенденції.

Для спрощення процесу створення Windows Forms додатків на мові C++ в середовищі Visual Studio 2022, був встановлений додатковий компонент - CppWinFormsProjectTemplate. Цей компонент призначений для забезпечення зручного початку роботи з проектами Windows Forms та сприяння ефективній розробці графічного інтерфейсу.

Після встановлення CppWinFormsProjectTemplate отримана можливість легко створювати нові проекти Windows Forms на мові C++. Використання цього шаблону суттєво спрощує початкову конфігурацію, оскільки він вже містить налаштовані параметри та готові шаблони. Це дозволяє прискорити та полегшити процес розробки графічного інтерфейсу для програмного додатку.

3.2.2 Створення клієнтської частини

Проведення розробки інтерфейсу користувача для введення вхідних даних є важливим етапом у створенні програмного додатку. На цьому етапі розробляється графічний інтерфейс, що дозволяє користувачеві зручно та ефективно вводити необхідну інформацію, та виконувати увесь цикл розрахункових робіт з переглядом результатів.

На першому етапі розробки інтерфейсу було створено основні вікна для роботи з різними режимами, що визначаються функціональністю програмного додатку. Ці вікна визначатимуть основні області взаємодії користувача з програмою, надаватимуть доступ до різних функцій та опцій, і будуть забезпечувати зручність управління та введення вхідних даних.

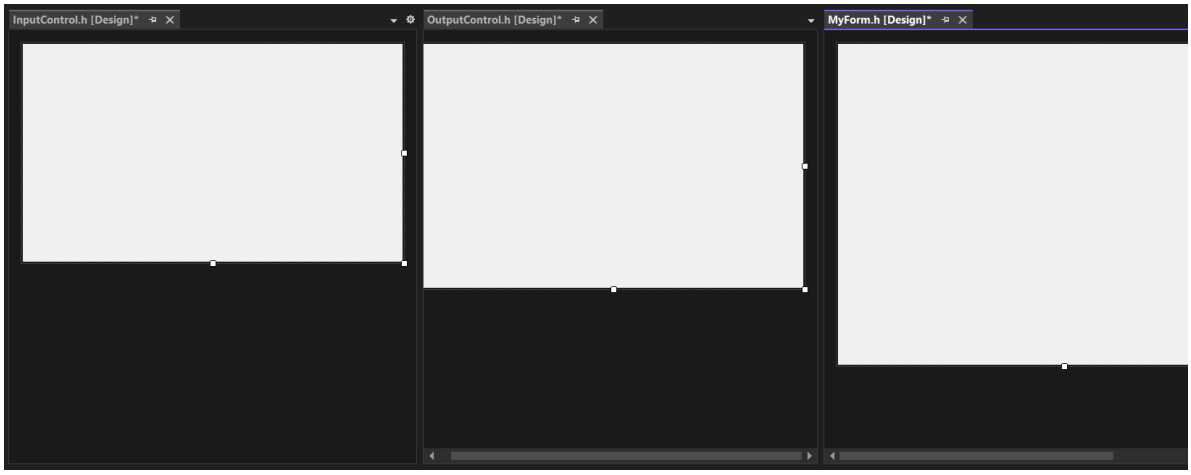


Рисунок 3.7 – Створення основних вікон програми

У процесі розробки програмного додатку використано такі елементи інтерфейсу користувача:

1. TextBox:

Тип елемента: Поле введення тексту.

Використання: Для введення числових значень, таких як тиск, температура, масовий потік і т.д.

2. Label:

Тип елемента: Текстова мітка.

Використання: Для відображення інструкцій користувачеві, позначення полів введення та відображення результатів обчислень.

3. Button:

Тип елемента: Кнопка.

Використання: Для запуску розрахунків і переключення між різними режимами роботи програми.

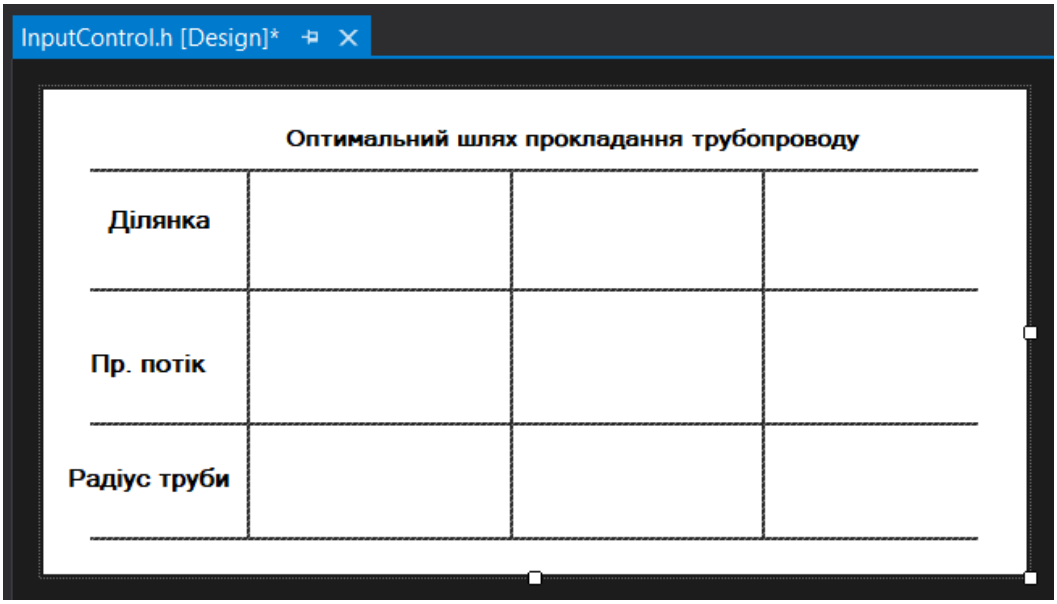


Рисунок 3.8 – Добавлення елементів на вікно виводу інформації

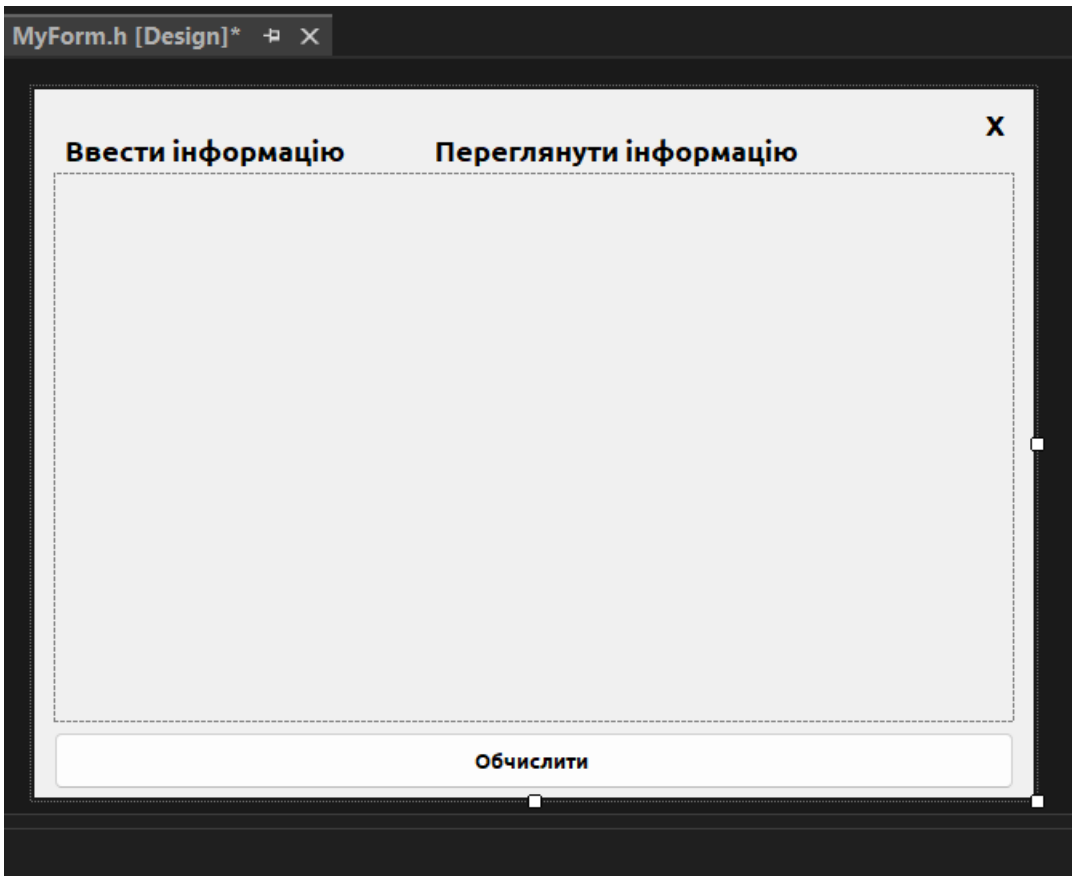


Рисунок 3.9 – Добавлення елементів на головне вікно

У вікні відображення даних з бази даних використано лише елемент керування DataGridView, який призначений для відображення та редагування даних у табличній формі. У контексті програми з термодинамікою, DataGridView дозволяє створювати табличний інтерфейс, де дані можуть бути представлені у вигляді рядків та стовпців.

DataGridView відображає дані з бази даних та надає можливості сортування, фільтрації та редагування інформації. Користувач може легко аналізувати результати обчислень газового потоку в трубі, використовуючи зручний табличний інтерфейс. Події, такі як вибір конкретного рядка чи стовпця, дозволяють забезпечити взаємодію з користувачем та забезпечити зручний доступ до інформації.

DataGridView є ефективним інструментом для роботи з даними у табличній формі, спрощуючи введення змін, відображення результатів та взаємодію з базою даних.

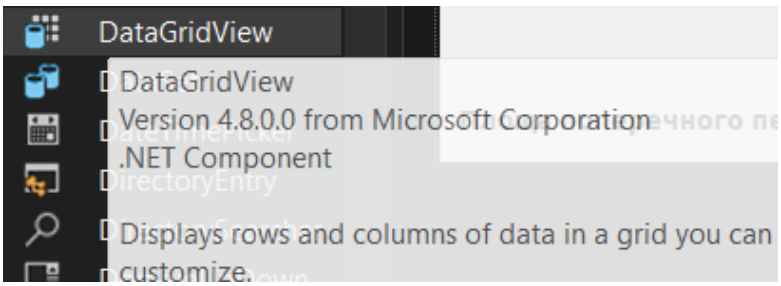


Рисунок 3.10 – Елемент DataGridView

При перегляді графічного дизайну програми вирішено, що існуючий варіант не влаштовує за вимогами та очікуваннями. З метою поліпшення користувацького інтерфейсу та візуальної привабливості було прийнято рішення про покращення графічного дизайну додатку. Це може включати в себе зміну кольорової палітри, розміщення елементів на формах, оптимізацію використання графічних елементів та удосконалення зручності використання для кінцевого користувача. Мета полягає в створенні естетичного та функціонального графічного інтерфейсу, який відповідає сучасним стандартам та вимогам користувачів.

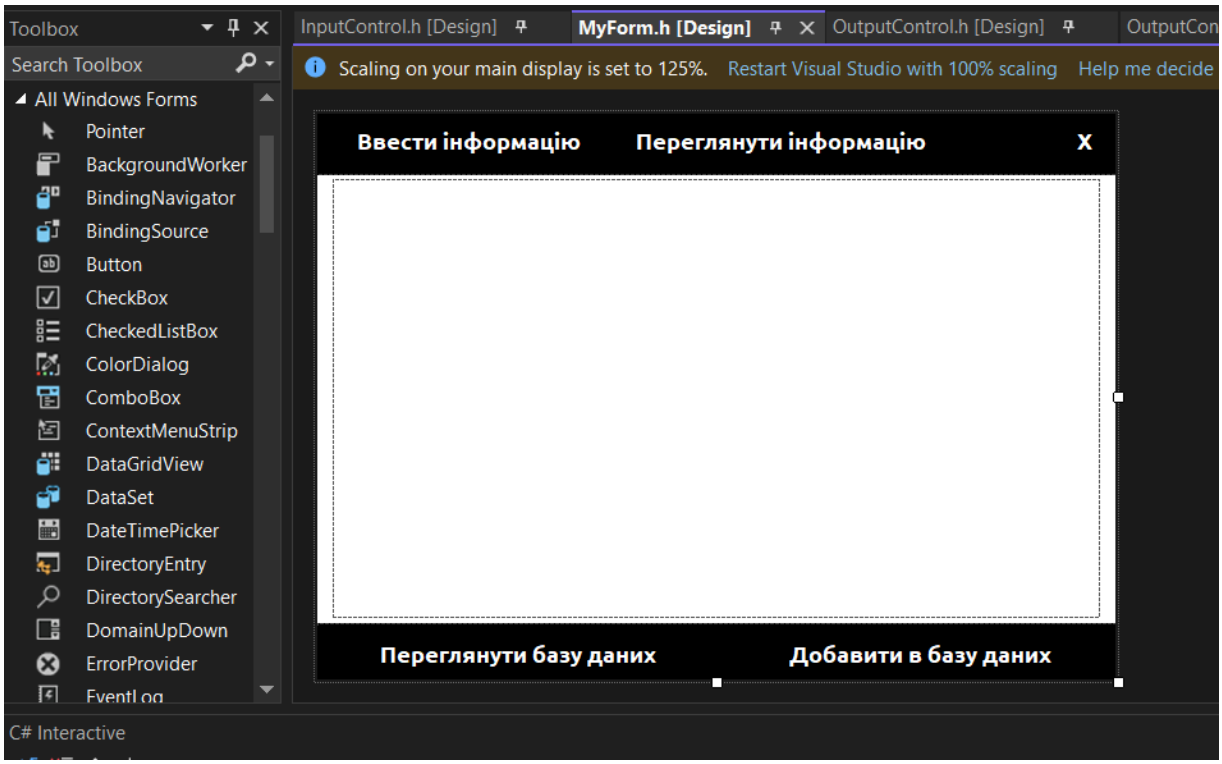


Рисунок 3.11 – Оновлений дизайн головного вікна

В результаті, після цього етапу отримується інтерфейс, який дозволяє користувачеві зручно та ефективно вводити вхідні дані для подальшого їх оброблення та використання у програмному додатку.

Далі розроблено клас Thermodynamics, який визначає різні термодинамічні параметри та обчислює їх значення. Визначені методи включають в себе установку параметрів, обчислення різних фізичних величин та повернення їх значень. Також даний клас вміщає розрахунок оптимального шляху прокладання трубопроводу реалізованим алгоритмом Беллмана-Форда, що враховує пропускні здатності та радіуси труб.

У конструкторі класу ініціалізуються різні змінні, такі як щільність, теплоємність, початковий тиск, початкова температура, радіус та площа.

Метод setParameters призначений для установки значень цих параметрів.

Метод calculate використовується для обчислення різних параметрів, таких як масовий витік, швидкість, тиск, температура, адіабатична ентальпія та інші.

Додатково, клас містить методи, які повертають значення розрахованих параметрів, такі як масовий витік, швидкість, тиск, температура, адіабатична ентальпія, ізентропічна ефективність, ізохоричний тиск, адіабатична температура.

Наприкінці класу подані реалізації функцій для обчислення цих параметрів. Наприклад, функції mass_flow, velocity, pressure, temperature, тощо, використовуються для розрахунків зазначених параметрів згідно зі зазначеними формулами.

Додавання обробників виняткових ситуацій є важливим аспектом розробки програмного додатку для забезпечення надійності та коректності його функціоналу. У програмі реалізовано обробники виняткових ситуацій, які використовуються для ефективної обробки непередбачуваних умов.

В одному з випадків, програма включає обробник, який перевіряє наявність значень у введених користувачем полях.

```
if (icontrol->CheckField()) {
    double density, specific_heat, initial_pressure,
        initial_temperature, radius, area, mass_flow_rate;
    if (Double::TryParse(icontrol->GetText1(), mass_flow_rate)
        && Double::TryParse(icontrol->GetText2(), radius)
        && Double::TryParse(icontrol->GetText3(), area))
    {
        thermodynamics->setParameters(density, specific_heat,
            initial_pressure, initial_temperature, radius, area);
        thermodynamics->calculate();

        MessageBox::Show("Внести дані");
    }
    else {
        MessageBox::Show("Введіть число!");
    }
}
else {
    MessageBox::Show("Не правильно введені дані!");
}
```

Рисунок 3.12 – Перевірка наявності введених даних

Якщо користувач намагається використати пусте поле або введені значення не є числом, обробник винятку виводить відповідне повідомлення користувачеві, сповіщаючи його про необхідність введення коректних даних.

Також, інші обробники виняткових ситуацій використовуються при виведенні обчислених термодинамічних процесів газового потоку в трубі. Ці обробники дозволяють уникнути виведення неочікуваних результатів, таких як "NaN" (Not a Number), якщо дані ще не були обчислені. Замість цього, вони елегантно керують ситуацією, де дані ще не готові для виведення, і приховують виведення до моменту завершення обчислень.

3.3 Використання програмного додатку

При запуску додатку користувач автоматично потрапляє на головний екран, де представлений інтерфейс для введення інформації. Це вікно забезпечує

користувача можливістю внесення необхідних даних для подальшого аналізу та обробки в програмі. Головний екран створений таким чином, щоб забезпечити зручність введення і відображення важливої інформації.

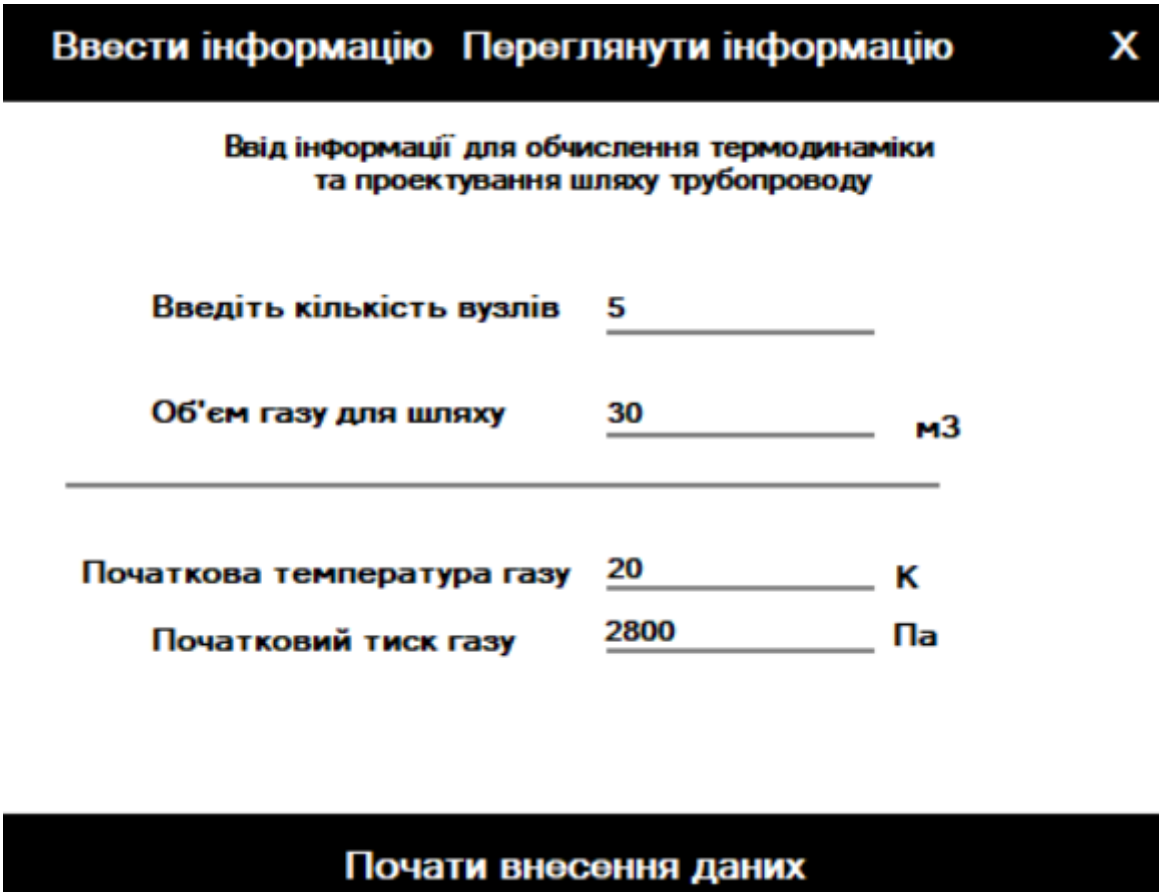


Рисунок 3.13 – Головна сторінка додатку

У програмі існують два режими роботи: "Введення інформації" та "Перегляд інформації". Кожен з них активується в головному вікні та виконує відповідні функції в зазначеній області.

Область для введення інформації включає поля, призначені для отримання користувальницьких даних.

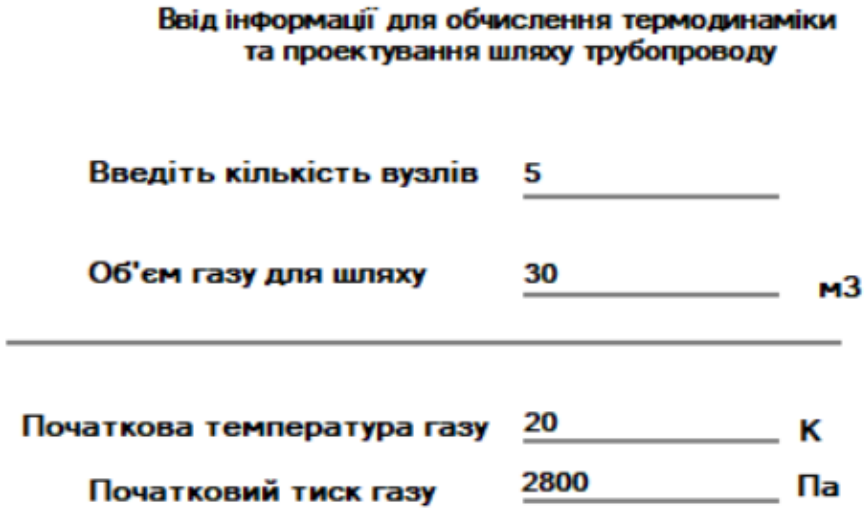


Рисунок 3.14 – Область введення інформації

В додатку присутня кнопка «Почати внесення даних», після натискання якої йде перевірка на те, чи поля пусті і чи введені дані є числами (цілими або дробовими). Якщо перевірка на виключні ситуації пройшла успішно, то система переходить до внесення необхідних даних.

Рисунок 3.15 – Кнопка «Внести дані»

Після успішного виконання обчислень термодинаміки газового потоку в трубі, користувач може переглянути всі відповідні дані в режимі "Перегляд інформації".

Рисунок 3.16 – Перегляд обчисленої інформації

Також в цьому режимі присутня кнопка «Переглянути базу даних», після нажимання на яку здійснюється підключення до бази даних, та запит на перегляд усіх записів.

Рисунок 3.17 – Перегляд списку з бази даних

Також під час нажимання на кнопку «Добавити в базу даних» під час успішної операції відображається вікно, яке сповіщає про це.

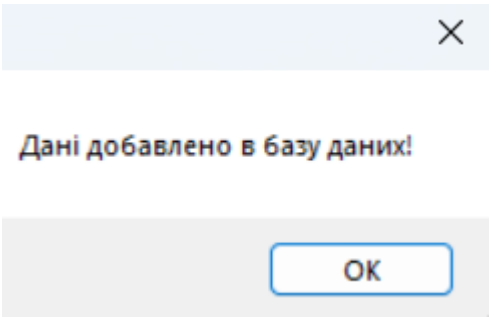


Рисунок 3.18 – Сповіщення про добавлення інформації в базу даних

3.4 Тестування програмного додатку

Процес тестування є ключовим етапом у розробці програмного продукту, спрямованим на забезпечення його якості та відповідності вимогам. Якісний продукт має задовольняти різноманітні сценарії використання, бути вільним від дефектів та володіти характеристиками надійності, безпеки, продуктивності,

зручності використання і розширюваності. Таким чином, тестування визначається як аналіз програмної системи для виявлення дефектів і оцінки її характеристик.

Процес тестування включає створення різних артефактів, таких як план тестування, який визначає стратегію тестування, ресурси та перелік тестів для кожної ітерації. Модель тестування охоплює контрольні завдання, методику випробувань, сценарії тестування, тестові скрипти і діаграми взаємодії тестів. Результати тестування використовуються для оцінки виконання тестів і виявлення дефектів.

Щодо характеристик користувацького інтерфейсу, важливо враховувати такі аспекти:

- Ясність: інтерфейс повинен бути зрозумілим і уникати двозначності, використовуючи відповідну мову, потік, ієрархію та метафори для візуальних елементів.
- Виразність: легкість створення зрозумілого інтерфейсу досягається завдяки уточненню та позначенню, уникайте перенасичення елементами для уникнення важкості використання.
- Знайомство: новий користувач повинен легко засвоювати інтерфейс, можливо, за допомогою реальних життєвих прикладів для передачі значень.
- Продуктивність: інтерфейс повинен робити користувача ефективним, сприяючи зручній взаємодії.

Система виконує перевірку на наявність порожніх полів у введених даних користувача. Якщо перевірка пройдена то система йде до наступної перевірки, якщо ні то виводиться помилка на екран.

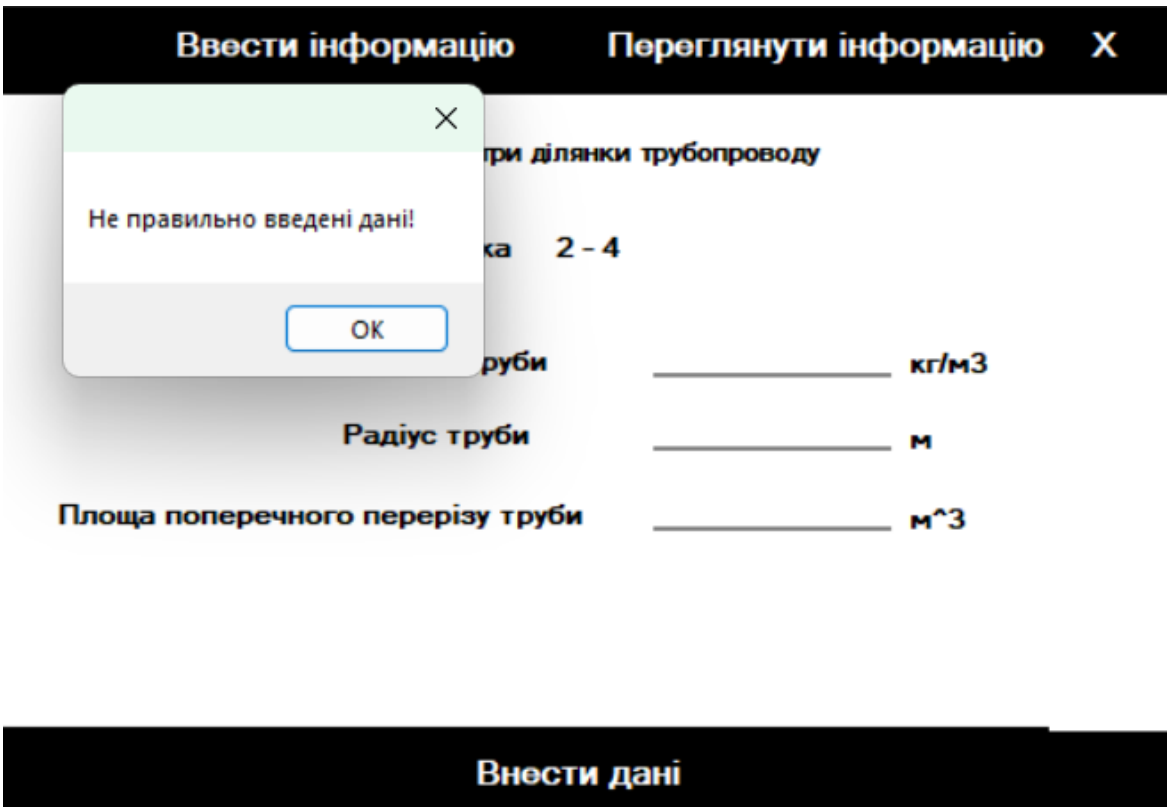


Рисунок 3.19 – Помилка при пустих полях

Після перевірки на пусті поля, йде перевірка чи введені дані являються числами.

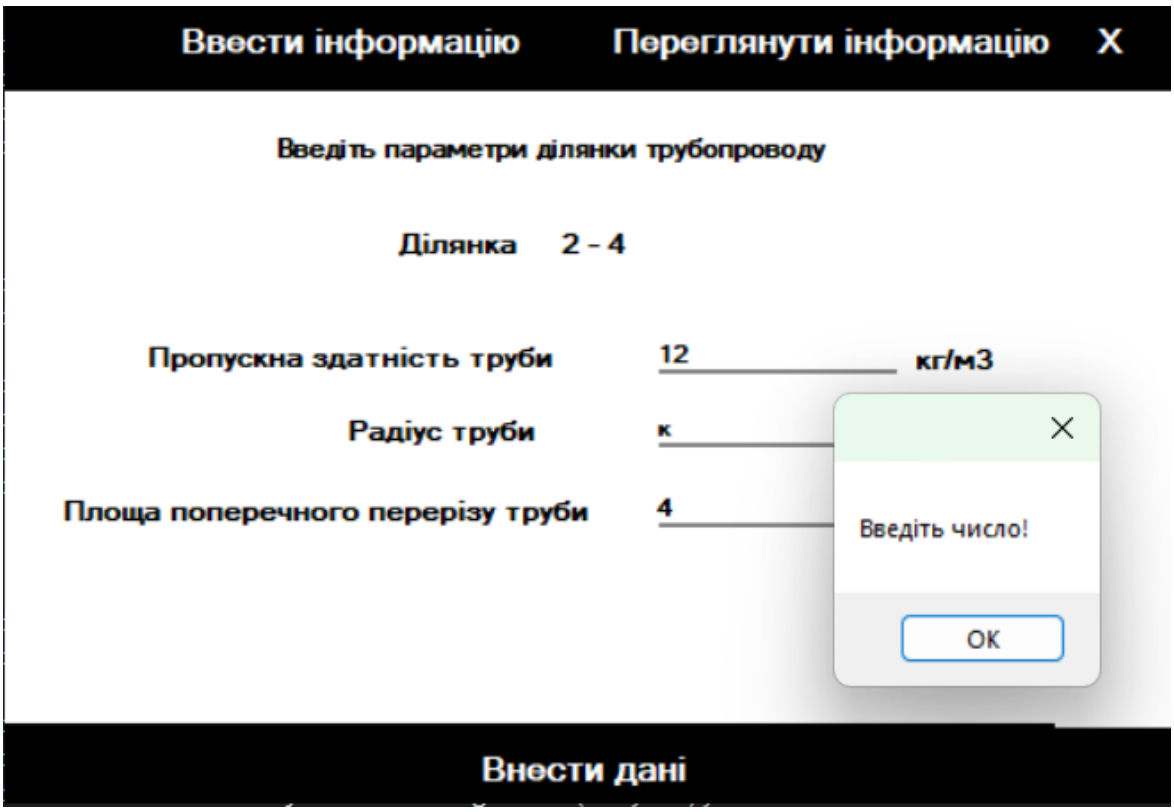


Рисунок 3.20 – Помилка при введені не вірних даних

Коли усі дані введено правильно, здійснюється обрахування.

Для перевірки коректності розрахованих даних використаємо онлайн калькулятор [30].

Даними для розрахунку початковими вважаємо значення використані користувачем у розробленому програмному додатку.

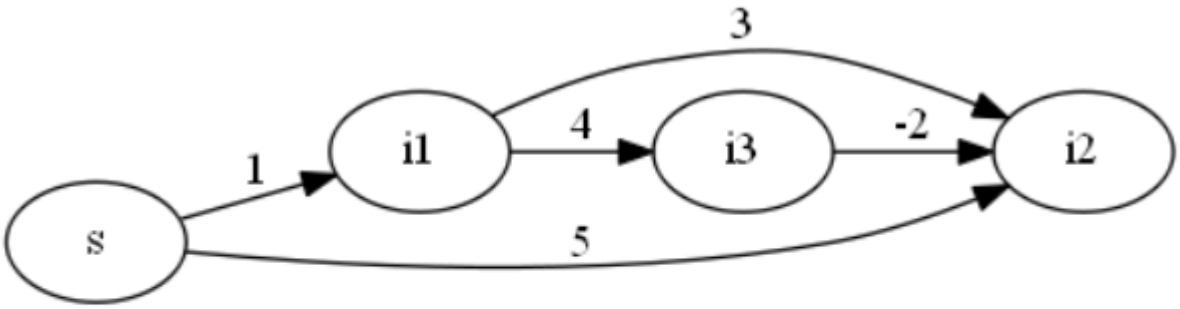
Вхідні дані:

– Точки: 1, 2, 3, 4, 5
– Пропускні здатності та радіуси труб (ділянка : пропуска здатність, радіус):
  - 1-2: 10 кг/м³, 20 м
  - 2-3: 8 кг/м³, 15 м
  - 3-5: 12 кг/м³, 10 м
  - 1-3: 15 кг/м³, 25 м
  - 1-4: 20 кг/м³, 30 м
  - 4-5: 10 кг/м³, 20 м
  - 2-4: 18 кг/м³, 22 м

- 2-5: 5 кг/м³, 12 м

– Обсяг газу для перекачування: 30 кг/м³

Задача полягає в тому, щоб знайти оптимальний шлях з точки 1 до точки 5, , який дозволить перекачати 30 кг/м³ за мінімальні затрати по часу транспортування.



Caluculate!!

Start from 1 .

Enter edges here!

```
1 2 10
2 3 8
3 5 12
1 3 15
1 4 20
4 5 10
2 4 18
2 5 5
```

Go

Рисунок 3.21 – Вхідні дані для обчислення в онлайн калькуляторі [30]

loop→ node↓	0	1	2	3	4	5
1	0(1)	0(1)	0(1)	0(1)	0(1)	0(1)
2	Infinity	10(1)	10(1)	10(1)	10(1)	10(1)
3	Infinity	15(1)	15(1)	15(1)	15(1)	15(1)
5	Infinity	Infinity	15(2)	15(2)	15(2)	15(2)
4	Infinity	20(1)	20(1)	20(1)	20(1)	20(1)

Рисунок 3.22 – Результати проведених онлайн обчислень

Обрахований результат надав найкоротші шляхи та їх ваги від обраної початкової вершини до всіх інших вершин, що дозволяє побачити, що оптимальний шлях з точки 1 до точки 5 через три ділянки труби з пропускними здатностями 10, 8 і 12 кг/м³, який відповідає знайденим програмним додатком значенням.

Рисунок 3.23 – Результат виконання задачі додатком

ВИСНОВКИ

У ході розробки програмного додатку проектування шляху прокладання трубопроводу природнього газу було проведено аналіз предметної області, визначено актуальні проблеми та оглянуто існуючі програмні рішення. Основною метою додатку є надання зручного інструмента для введення та відстеження термодинамічних даних.

Моделювання та проектування додатку включало в себе створення відповідної архітектури та моделі бази даних для забезпечення ефективного зберігання і управління інформацією. У процесі програмної реалізації було використано класи.

Важливою частиною було виявлення класів сутності, що стало основою для подальшого проектування програмного забезпечення. Цей етап визначив ключові компоненти системи та їх взаємозв'язки, що дало можливість розробити більш детальний план програмного продукту.

На етапі проектування програмного забезпечення було визначено ключові класи системи, що визначають основні компоненти програмного продукту. Кожен клас був ретельно проаналізований, а його функціонал був детально описаний для забезпечення повноти та точності реалізації.

Додаток успішно реалізовано і відповідає вимогам, що зазначені у технічному завданні. Введені дані піддаються перевірці на коректність, а в разі успішного обчислення термодинаміки газового потоку результати можуть бути переглянуті у режимі "Перегляд інформації".

Загалом, розроблений програмний додаток є ефективним інструментом для вирішення задач з обчислення оптимального шляху прокладання трубопроводу і розрахунку термодинамічних параметрів газового потоку в трубі, та відображення результатів користувачу.

Розширення функціоналу та покращення дизайну інтерфейсу стануть пріоритетними завданнями для подальшого вдосконалення додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. White, F. M. Fluid Mechanics. McGraw-Hill, 2016. 768 с.
2. Sussman, M., Smereka, P. Axisymmetric Free Surface Flows: A Level Set Approach // Journal of Computational Physics. 1994. № 114(1). С. 146-159
3. Roy, C. J., Oberkampf, W. L. A Comprehensive Framework for Verification, Validation, and Uncertainty Quantification in Scientific Computing // Computers & Fluids. 2014. № 90. С. 20-29.
4. Халатов А.А., Гільчук А.В., Кохтич Л.М. Термодинаміка газового потоку в трубі – Київ: КПІ ім. Ігоря Сікорського, 2019. – 219 с.
5. Гільчук А.В., Панченко Н.А., Мейріс А.Ж. Методичний посібник для практичних занять з курсу «Термодинаміка газового потоку». – Київ: КПІ ім. Ігоря Сікорського, 2017. – 71 с.
6. The Pipeline Simulation Interest Group (PSIG): 54th ANNUAL CONFERENCE [Електронний ресурс] – Режим доступу до ресурсу: https://www.simonecongress.com/2024/ (дата звернення: 13.05.2024)
7. SIMONE Congress 2024 [Електронний ресурс] – Режим доступу до ресурсу: https://www.simonecongress.com/2024/ (дата звернення: 13.05.2024)
8. OpenFOAM [Електронний ресурс] – Режим доступу до ресурсу: https://www.openfoam.com/ (дата звернення: 14.05.2024)
9. The COMSOL Multiphysics [Електронний ресурс] – Режим доступу до ресурсу: https://www.comsol.com/ (дата звернення: 14.05.2024)
10. Ansys Fluent Fluid Simulation Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.ansys.com/products/fluids/ansys-fluent (дата звернення: 14.05.2024)
11. SimScale - High-fidelity engineering simulation on any hardware [Електронний ресурс] – Режим доступу до ресурсу: https://www.simscale.com/ (дата звернення: 14.05.2024)

12. Наукове моделювання | Science Learning Hub [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencelearn.org.nz/resources/575-scientific-modelling (дата звернення: 15.05.2024)
13. Dubois, Paul. MySQL Cookbook. O'Reilly Media, 2014. 974 с.
14. Create a Windows Forms app in Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022 (дата звернення: 17.05.2024)
15. IdeaScale. What is the Use Case Diagram? Definition, Uses, Examples, and UML Use Case Diagram [Електронний ресурс] – Режим доступу: https://www.ideascale.com/what-is-the-use-case-diagram/ (дата звернення: 19.05.2024)
16. Що таке C++ [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/C%2B%2B
17. Learn C++ Programming [Електронний ресурс] – Режим доступу до ресурсу: https://www.programiz.com/cpp-programming - google_vignette
18. Посібник C++ [Електронний ресурс] – Режим доступу до ресурсу: https://repository.kpi.kharkov.ua/bitstream/KhPI-Press/54410/1/Book_2021_Sobol_Osnovy_prohramuvannia.pdf
19. Introduction to object-oriented programming [Електронний ресурс] – Режим доступу до ресурсу: https://www.learncpp.com/cpp-tutorial/introduction-to-object-oriented-programming/
20. C++ Object Oriented [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/cplusplus/cpp_object_oriented.htm
21. CodeAcademy, C++ [Електронний ресурс] – Режим доступу до ресурсу: https://www.codecademy.com/learn/c-plus-plus-for-programmers/modules/object-oriented-programming-in-cpp/cheatsheet
22. Desktop Guide [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-8.0

23. MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysql.com/
24. What Is MySQL and How Does It Work [Електронний ресурс] – Режим доступу до ресурсу: https://www.hostinger.com/tutorials/what-is-mysql
25. MySQL Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/mysql/
26. Workbeanch [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysql.com/products/workbench/
27. Software Modeling and Design [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/okuuz
28. UML, Use Cases, Patterns, and Software Architectures [Електронний ресурс] – Режим доступу до ресурсу: http://surl.li/okuup
29. CppWinFormsProjectTemplate [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/thisismalindu/CppWinFormsProjectTemplate?tab=readme-ov-file
30. Bellmanford calculator [Електронний ресурс] – Режим доступу до ресурсу: https://heabiside.github.io/calculators/bellmanford/bellmanford.html
31. Bellman-Ford Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://brilliant.org/wiki/bellman-ford-algorithm/

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи
«Програмний додаток проектування шляху
прокладання трубопроводу природнього газу»

ПОГОДЖЕНО:

В. о. завідувача кафедри

_____ Ващенко С.М.

Студентка групи ІТ-01__

_____ Недайхліб М.С.

Суми 2024

1. Призначення й мета створення програмного додатку

1.1 Призначення програмного додатку

Інформаційна система має забезпечить користувачеві можливість детального аналізу та ефективного розрахунку показників термодинамічних процесів в газових потоках для прокладання нових газопроводів та аналізу існуючих шляхів газотранспортних мереж.

1.2 Мета створення програмного додатку

Спростити та прискорити аналіз термодинамічних процесів газового потоку в трубі, забезпечуючи точні та достовірні результати для подальшої перспективи прокладання газопроводу. З урахуванням практичного застосування отриманих даних, додаток відкриває нові перспективи для вдосконалення технічних рішень та забезпечення оптимальної ефективності газотранспортних систем та інших інженерних проектів.

1.3 Цільова аудиторія

До цільової аудиторії програмного додатку можна віднести інженерів, науковців та фахівців в області термодинаміки газів, які зможуть застосувати отримані дані як нові перспективи для вдосконалення технічних рішень та забезпечення оптимальної ефективності газотранспортних систем, прокладання нових газопроводів та редизайн існуючих трубних мереж, та інших інженерних проектів.

2 Вимоги до програмного додатку

2.1 Вимоги до програмного додатку в цілому

2.1.1 Вимоги до структури й функціонування програмного додатку

Програмний додаток має бути доступним для використання працівниками промисловості та сфери наукових досліджень. Програмний додаток повинен складатися із взаємозалежних компонентів використання системи.

2.1.2 Вимоги до персоналу

Окрім базових знань роботи з електронними таблицями, персонал повинен володіти загальними навичками роботи з ПК, досвідом роботи з програмним забезпеченням і мінімальними технічними знаннями для обслуговування та роботи з програмним забезпеченням.

2.1.3 Вимоги до збереження інформації

Базу даних програмного продукту для зберігання обчислених параметрів та отримання збережених даних реалізовано засобами системи управління базами даних MySQL.

2.1.4 Вимоги до розмежування доступу

Додаток має бути доступний для завантаження або купівлі через відповідні дистрибутивні платформи або прямо з веб-сайту розробника.

Всі користувачі поділяються на дві основні категорії: звичайні користувачі (клієнти) та адміністратор.

Клієнти мають доступ до головної частини додатку – обчислення та база даних результатів, що включатиме різні можливості введення даних, отримання результатів, та інші ключові функції, спрямовані на задоволення потреб користувача в аналізі термодинамічних процесів газового потоку в трубі.

Адміністратор може вносити зміни в інтерфейс додатку та налаштування системи .

2.2 Структура програмного додатку

2.2.1 Загальна інформація про структуру програмного додатку

Структура програмного додатку являє цілісну структуру взаємопов’язаних класів, які також є пунктами головного меню:

Головний екран створений таким чином, щоб забезпечити зручність введення і відображення важливої інформації.

У програмі існують два режими роботи: "Введення інформації" та "Перегляд інформації". Кожен з них активується в головному вікні та виконує відповідні функції в зазначеній області.

Пункт меню «Переглянути базу даних» відображає спеціальне вікно зі списком записів з бази даних.

Після успішного виконання обчислень термодинаміки газового потоку в трубі, користувач може переглянути всі відповідні дані в режимі "Перегляд інформації" та виконати

2.2.2 Навігація

Користувач входить в систему, після чого йому відображається головне вікно з областю для вводу потрібної інформації для обрахування. Користувач заповнює поля з даними та натискає на кнопку обрахувати, тоді може перейти на перегляд другої області з обрахованою термодинамікою газового потоку.

2.2.3 Наповнення програмного додатку (вхідні дані)

Заповнення та редагування вхідних даних має бути зроблено через відповідний режим головного меню. Дана клієнтська частина повинна мати інтерактивний інтерфейс для введення похідних параметрів, таких як густини газу, радіус труби, початковий тиск, тощо.

Інформацію для введення даних користувач бере з документації експлуатації газорозподільчих мереж, технічної звітності, інших інформаційних систем, тощо.

2.2.4 Дизайн та структура додатку

Легкість створення дизайну додатку досягається завдяки уточненню та позначенню, уникаючи перенасичення елементами для запобіганню важкості використання.

Структура програмного додатку має бути зрозумілою і уникати двозначності, використовуючи відповідну мову, потік, ієрархію та метафори для візуальних елементів.

Розташування елементів на головній сторінці програмного додатку схематично показано на рисунку А.1.

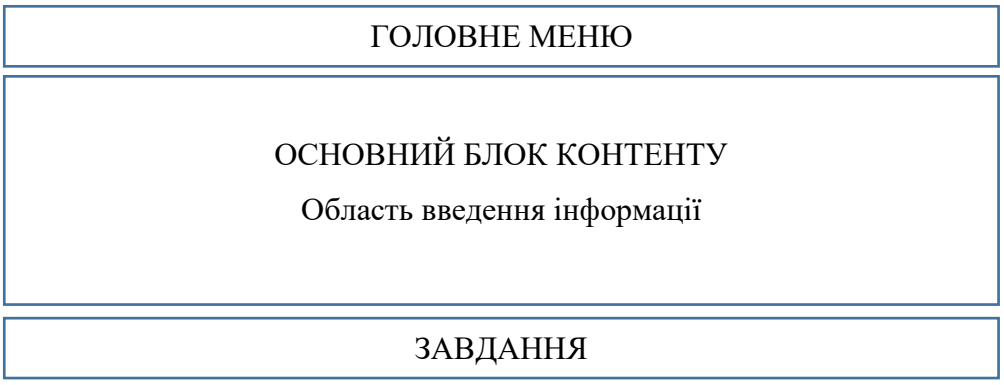


Рисунок А.1 – Схема головного екрану додатку

Макет подальшого екрану додатку наведено нижче.

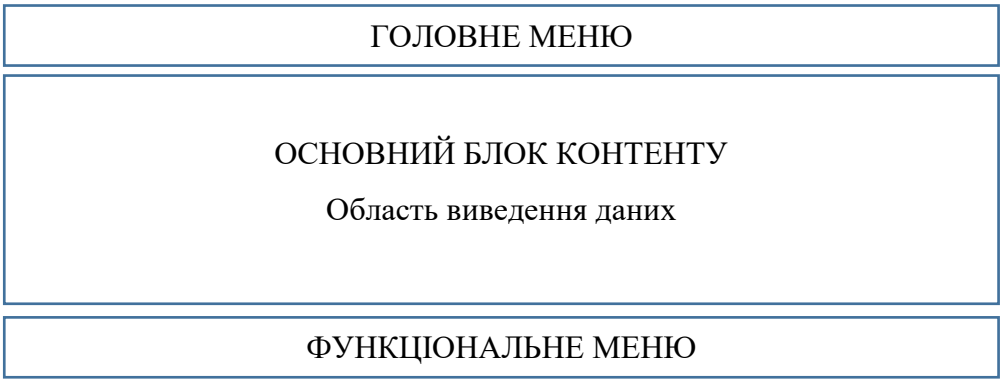


Рисунок А.2 – Схема екрану додатку режиму виведення інформації

2.2.5 Система навігації (карта програмного додатку)

Карта програмного додатку зображена на рисунку А.2.

Рисунок А.2 – Карта додатку

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	Потреби користувача	Джерело
UN-01	Перегляд сторінки внесення інформації	Клієнт
UN-02	Перегляд сторінки з обрахуванням	Клієнт
UN-03	Внесення інформації	Клієнт

Продовження таблиці А.1

UN-04	Обчислення вхідних даних	Клієнт
UN-05	Додати в базу даних результати	Клієнт
UN-06	Переглянути базу даних результатів	Клієнт

2.3.2 Функціональні вимоги

На основі потреб користувача були визначені такі функціональні вимоги:

- Визначення температури, тиску та об'єму газу за різних умов.
- Розрахунок кількості роботи та внутрішньої енергії газу.
- Вивід результатів енергетичних змін та температурних показників.
- Реалізація обчислень для ентропії та ентальпії під час термодинамічних процесів.
- Обчислення теплоємності та інших характеристик речовини.
- Реалізація можливості введення різних умов та параметрів для аналізу різних термодинамічних сценаріїв.

2.3.3 Системні вимоги

Даний розділ визначає, розподіляє та вказує на системні вимоги, визначені розробником. Їх перелік наведений в таблиці А.2.

Таблиця А.2 – Системні вимоги

ID	Системні вимоги	Пріоритет	Опис
SR-01	Модуль введення даних	M	Надає можливість клієнту внесення вхідних даних
SR-02	Обчислення	M	Проводить обчислення згідно заданих значень параметрів

Продовження табл. А.2.

SR-03	Модуль перегляду результатів	M	Надає можливість переглянути результати обчислень
SR-04	База даних результатів	M	Надає можливість відображати зберігати та переглядати вже наявні результати певних обчислень
SR-05	Зміна системи вимірювання	C	Відповідає за одиниці вимірювання для введених даних (метрична/імперіальна)

Умовні позначення в таблиці А.2:

Must have (M) – вимоги, які повинні бути реалізовані в системі;

Could have (C) – вимоги, які можуть бути реалізовані, але вони не є центральною ціллю проекту.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Впровадження програмного додатку досягається з використанням:

- CppWinFormsProjectTemplate
- C++
- MySQL

2.4.2 Вимоги до лінгвістичного забезпечення

Програмний додаток має бути виконаний українською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення передбачає наявність операційної системи Windows версії 7 і більше та вільний об’єм пам’яті жорсткого диску не менше 1 ГБ.

3 Склад і зміст робіт зі створення програмного додатку

Докладний опис етапів роботи зі створення програмного додатку наведено в таблиці А.3.

Таблиця А.3 – Етапи створення програмного додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Підготовка специфікації	9 днів
1.1	Аналіз предметної області	4 дні
1.1.1	Визначення проблеми які, вирішить додаток	2 день
1.1.2	Аналіз аналогів програмних додатків	2 дні
1.2	Визначення властивостей програмного додатку	5 днів
1.2.1	Розробка ТЗ	1 дні
1.2.2	Редагування та обговорення ТЗ	3 дні
1.2.3	Затвердження ТЗ	1 день
2	Розробка програмного додатку	51 день
2.1	Front-end розробка	18 днів
2.1.1	Розробка шаблону додатку	3 дні
2.1.2	Верстка екранів додатку	15 днів
2.2	Back-end розробка	24 дні
2.2.1	Розробка модулю введення інформації	6 днів
2.2.2	Розробка модулю обчислення	12 днів
2.2.**3**	Розробка модулю виведення результатів	6 днів
2.3	Розробка та підключення бази даних результатів	9 днів
3	Тестування програмного додатку	14 днів

Продовження таблиці А.3.

3.1	Бета-тестування	9 днів
3.2	Альфа-тестування	5 днів
4	Введення в експлуатацію	10 днів
4.1	Налагодження коректної роботи додатку	6 днів
4.2	Розміщення додатку на відповідній платформі	1 день
4.3	Написання супровідної документації	2 дні
4.4	Реліз програмного додатку	1 день

4 Вимоги до складу й змісту робіт із введення програмного додатку в експлуатацію

Щоб програмний додаток міг бути доступним для клієнтів, його необхідно розмістити для завантаження на відповідних платформах або на веб-сайті розробника. Перед цим потрібно забезпечити пакування додатку в інсталяційні пакети відповідно до вимог операційних систем користувачів. Також слід встановити захист і шифрування для забезпечення безпеки додатку та даних користувачів. Для успішного розгортання та оновлення додатку на користувацьких системах необхідно, щоб параметри системи користувачів відповідали технічним вимогам, зазначеним у ТЗ.

ДОДАТОК Б

Планування робіт

Деталізація мети проекту методом SMART. Для успішності та конкурентоспроможності проєкту треба на концептуальному етапі правильно визначити його мету за допомогою SMART-методу. "Розробити програмний додаток проектування шляху прокладання трубопроводу природнього газу мовою програмування C++, досягнути точності обчислень не менше 95%, та завершити розробку згідно з погодженим календарним планом виконання робіт.".

Результати деталізації методом SMART розміщені у табл. Б.1.

Таблиця Б.1 − Деталізація мети методом SMART

Specific (конкретна)	Розробити програмний додаток проектування шляху прокладання трубопроводу природнього газу мовою програмування C++
Measurable (вимірювана)	Досягнути точності обчислень не менше 95%
Achievable (досяжна)	Реалізація додатку виконується мовою програмування C++, використовуючи компонент CppWinFormsProjectTemplate
Relevant (реалістична)	Додаток, який відповідає потребам наукових досліджень, промислових та інженерних розрахунків, покращуючи точність та ефективність термодинамічних обчислень
Time-framed (обмежена у часі)	Завершення розробки додатку має бути згідно з погодженим календарним планом.

Планування змісту структури робіт. WBS (Work Breakdown Structure – ієрархічна структура робіт) – це графічний вигляд елементів проєкту, які згруповані ієрархією у єдине ціле з продуктом проєкту. Структура декомпозиції робіт орієнтована на досконале виконання робіт по частинам і сама є ключовою частиною проєкту, яка спрямована на організацію командної роботи. Елементами декомпозиції можуть бути продукти, дані та послуги. Більше того, WBS забезпечує необхідним каркасом для ретельної оцінки термінів та контролю та графіків роботи. На найвищому (першому) рівні розміщений продукт проєкту. Основні дії та заходи, що забезпечують досягнення мети проєкту, зафіксовані на другому рівні декомпозиції. Декомпозиція робіт виконується до тих пір, поки вони не стануть елементарними (простими). Елементарні роботи – це дії, які мають однозначний чіткий результат, на які призначена відповідальному одна конкретна особа, для якої можна обчислити витрати праці і тривалість виконання. Діаграма WBS зображена на Рисунок Б.1.

Планування структури організації, для впровадження готового проекту (OBS). Наступним етапом після декомпозиції процесів є розробка організаційної структури виконавців або OBS, яка визначається як графічна структура відображення учасників або відповідальних осіб, які беруть участь у реалізації проєкту. У ролі відповідальних осіб виступають співробітники, що відповідають за організацію і виконання елементарної роботи, що зазначена у WBS. Кожну елементарну роботи можна розглядати як окремий проєкт. Діаграма OBS зображена на Рисунок Б.2. Список виконавців, що функціонують в проекті знаходиться в табл. Б.2.

Таблиця Б.2 − Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Недайхліб М.С.	Виконує front-end та back-end розробку
Проектувальник	Недайхліб М.С.	Виконує проєктування бази даних та розробляє структуру програмного додатку.

Продовження таблиці Б.2

Тестувальник	Недайхліб М.С.	Відповідає за тестування функціоналу та дизайну програмного додатку
Косультант проекту	Ващенко С.М.	Формує завдання на розробку проєкту.
Менеджер проекту	Недайхліб М.С.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Виконує збір та аналіз даних.

Рисунок Б.1 − WBS. Структура робіт проекту

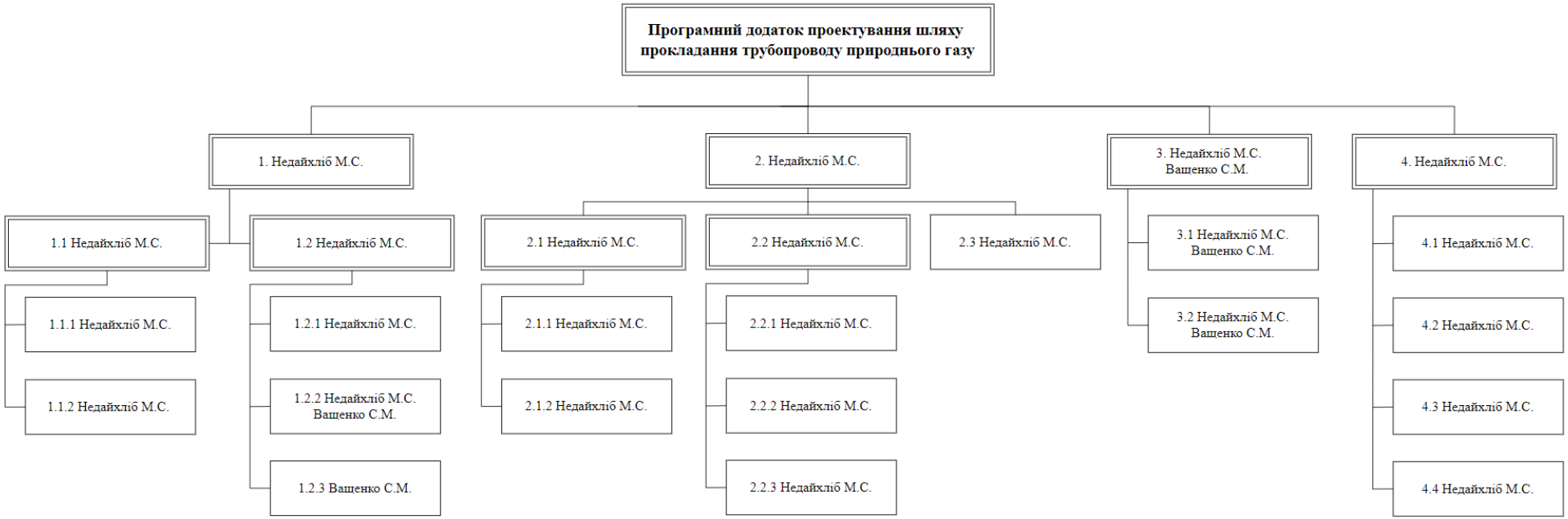


Рисунок Б.2 − Організаційна структура проекту (OBS)

Діаграма Ганта. Далі побудуємо календарний план виконання дипломного проекту. Найпоширеніший формат графіка в будь-якій галузі — діаграма Ганта. Побудова календарного графіку (діаграми Ганта) є одним з важливих етапів планування проєкту, що виглядає як розклад виконання робіт з реальним розподілом дат. Завдяки йому можна отримати достовірне уявлення про тривалість процесів з обмеженнями у ресурсах, урахуванням вихідних днів та свят. Діаграма Ганта та список робіт діаграми Ганта зображені на Рисунок Б.3-Б.4.

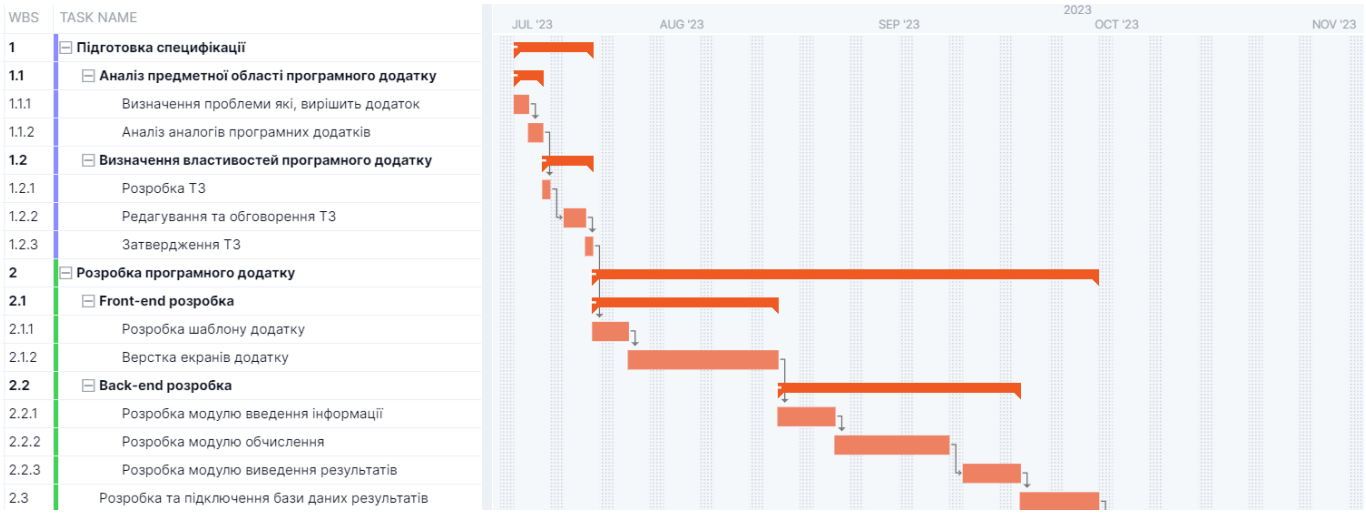


Рисунок Б.3 – Діаграма Ганта

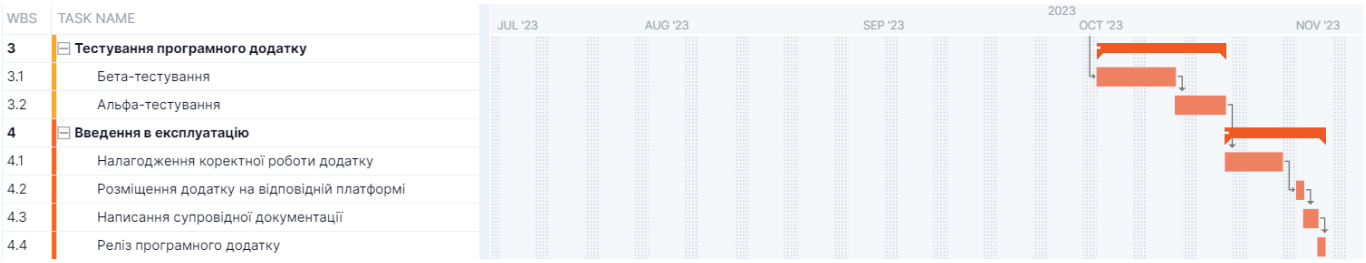


Рисунок Б.4 – Продовження діаграми Ганта

WBS	TASK NAME	DURATION	BASELINE STA...	BASELINE FINI...
1	**Підготовка специфікації**	**9 days**	**7/24/2023**	**8/3/2023**
1.1	**Аналіз предметної області програмного додатку**	**4 days**	**7/24/2023**	**7/27/2023**
1.1.1	Визначення проблеми які, вирішить додаток	2 days	7/24/2023	7/25/2023
1.1.2	Аналіз аналогів програмних додатків	2 days	7/26/2023	7/27/2023
1.2	**Визначення властивостей програмного додатку**	**5 days**	**7/28/2023**	**8/3/2023**
1.2.1	Розробка ТЗ	1 day	7/28/2023	7/28/2023
1.2.2	Редагування та обговорення ТЗ	3 days	7/31/2023	8/2/2023
1.2.3	Затвердження ТЗ	1 day	8/3/2023	8/3/2023
2	**Розробка програмного додатку**	**51 days**	**8/4/2023**	**10/31/2023**
2.1	**Front-end розробка**	**18 days**	**8/25/2023**	**9/14/2023**
2.1.1	Розробка шаблону додатку	3 days	8/4/2023	8/24/2023
2.1.2	Верстка екранів додатку	15 days	8/25/2023	9/14/2023
2.2	**Back-end розробка**	**24 days**	**9/15/2023**	**10/18/2023**
2.2.1	Розробка модулю введення інформації	6 days	9/15/2023	9/22/2023
2.2.2	Розробка модулю обчислення	12 days	9/25/2023	10/10/2023
2.2.3	Розробка модулю виведення результатів	6 days	10/11/2023	10/18/2023
2.3	Розробка та підключення бази даних результатів	9 days	10/19/2023	10/31/2023
3	**Тестування програмного додатку**	**14 days**	**11/1/2023**	**11/20/2023**
3.1	Бета-тестування	9 days	11/1/2023	11/13/2023
3.2	Альфа-тестування	5 days	11/14/2023	11/20/2023
4	**Введення в експлуатацію**	**10 days**	**11/21/2023**	**12/4/2023**
4.1	Налагодження коректної роботи додатку	6 days	11/21/2023	11/28/2023
4.2	Розміщення додатку на відповідній платформі	1 day	11/29/2023	11/29/2023
4.3	Написання супровідної документації	2 days	11/30/2023	12/1/2023
4.4	Реліз програмного додатку	1 day	12/4/2023	12/4/2023

Рисунок Б.5 – Список робіт для побудови діаграми Ганта

Аналіз ризиків. Виконаємо якісну і кількісну оцінку ризиків роботи. При якісній оцінці визначимо ризики, що потребують швидкого реагування. Така оцінка визначить ступінь важливості ризику і дозволить вибрати спосіб реагування. Кількісна оцінка ризиків буде виконана для більш повної ідентифікації ризиків та ступеня їхнього впливу на виконання проекту. Кількісна і якісна оцінка ризиків можуть використовуватися окремо або разом, залежно від наявного часу і бюджету, необхідності в кількісній або якісній оцінці ризиків. У табл. Б.5 знаходиться класифікація ризиків за показниками ймовірності виникнення ризику та величині втрат.

Далі виконаємо планування реагування на ризики — це розробка методів і технологій зниження негативного впливу ризиків на проект. Визначимо ефективність розробки реагування на проект, визначимо чи будуть наслідки впливу ризику на проект позитивними або негативним. Оцінюємо ризики за показниками, що знаходяться в табл. Б.3. На основі оцінки будуємо матрицю ймовірності виникнення ризиків та впливу ризику, що зображена на Рисунок Б.7.

Таблиця Б.3 − Шкала оцінювання ймовірності виникнення та впливу ризику на виконання проекту

Оцінка	Ймовірність виникнення	Вплив ризику
1	Низька	Низький
2	Середня	Середній
3	Висока	Високий

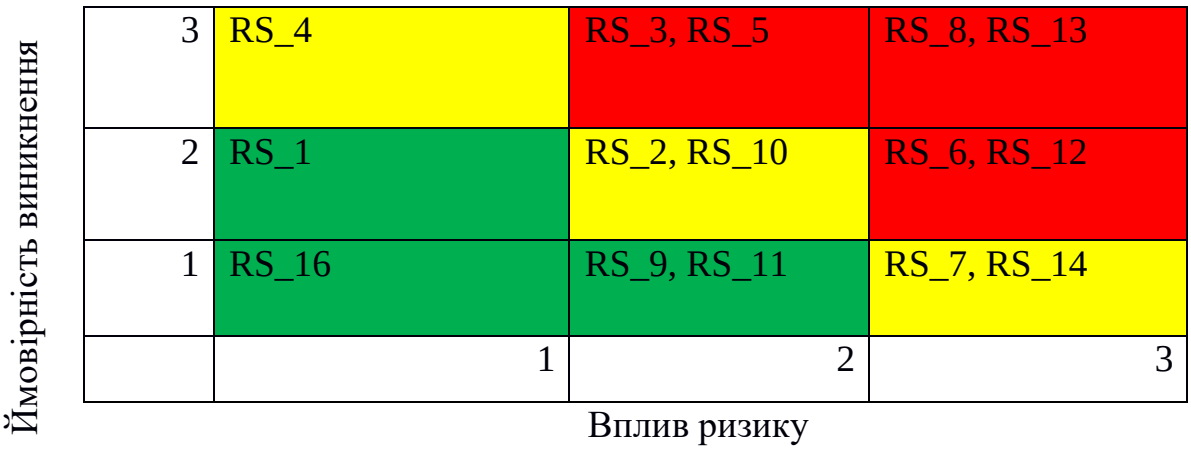


Рисунок Б.7 − Матриця ймовірності виникнення ризиків та впливу ризику

- зелений колір – прийнятні ризики;
- жовтий колір – виправданні ризики;
- червоний колір – недопустимі ризики.

На підставі отриманого значення індексу ризики класифікують: за рівнем ризику, що знаходиться в табл. А.4.

Таблиця Б.4 − Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	1≤R≤2	1,9,11,16
2	Виправдані	3≤R≤4	2,4,7,10,14
3	Недопустимі	6≤R≤9	3,5,6,8,12,13

Таблиця Б.5 − Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Несприйняття між розробником та клієнтом	Низька	Середній	3	1. Сприяти розвитку позитивних взаємин між розробником та клієнтом. 2. Відповідати стандартам етикету у професійному спілкуванні. 3. Створювати оптимальне середовище для співпраці.	Попередження	У разі виникнення розбіжностей важливо встановити основні причини недорозумінь, обговорити їх та спрямувати зусилля на підтримку здорової атмосфери у команді.

Продовження таблиці Б.5

RS_2	Відкритий	Обмежені професійні навички у команди розробників	Середня	Середній	4	1. Забезпечити професійний розвиток співробітників. 2. Використовувати інтернет-ресурси для покращення кваліфікації..	Пом’якшення	Запланувати час для навчання працівників, забезпечити їх необхідними книгами та організувати доступ до онлайн-курсів.
RS_3	Відкритий	Зміни у законодавстві	Середня	Високий	6	1. Періодичний огляд законодавчих змін та юридичний аудит. 2. Підготовка резервного плану для швидкої адаптації додатку до нових вимог.	Пом’якшення	Розробка процедури швидкого оновлення додатку, включаючи технічні, юридичні та оперативні заходи для відповідності новому законодавству.
RS_4	Відкритий	З'явлення конкурентного продукту	Низька	Високий	3	1. Здійснити аналіз ринку для оцінки конкуруючих продуктів. 2. Розробити унікальний підхід до створення проекту.	Прийняття	

Продовження таблиці Б.5

RS_5	Відкритий	Недостатня деталізація проектного завдання	Середня	Високий	6	1. Детально обговорити з замовником всі вимоги до проекту. 2. Розробити глосарій термінів, щоб уникнути непорозумінь у тлумаченні. 3. Регулярний перегляд замовником прогресу в роботі.	Попередження	У разі виявлення розбіжностей між реальними параметрами продукту і вимогами, необхідно чітко визначити та виправити помилки у виконанні.
RS_6	Відкритий	Проблеми з програмним забезпеченням у користувачів	Високий	Середній	6	Проектування з урахуванням специфічних вимог до програмного забезпечення користувачів.	Прийняття	Адаптація проекту до різних версій програмного забезпечення, яке використовуватиметься
RS_7	Відкритий	Зміни вимог замовника під час розробки проекту	Низька	Високий	3	Забезпечити повне узгодження усіх аспектів на початку проекту для зниження кількості змін у процесі.	Пом’якшення	Переглядати і оновлювати проект з кожною зміною вимог.

Продовження таблиці Б.5

RS_8	Відкритий	Недостатньо ефективний розподіл часу	Висока	Високий	9	Аналізувати ключові аспекти та завдання для оптимального розподілу часу. Підкреслити важливість точного дотримання графіка.	Пом’якшення	Переоцінка пріоритетів завдань і розгляд методів оптимізації існуючих планів. Зустрічі з замовником щодо можливих корекцій термінів виконання
RS_9	Відкритий	Неправильне уявлення про обсяг проекту	Середня	Низький	2	1. Проведедення грунтовного аналізу проекту, ідентифікація ключових фаз та розробка часового графіку для кожної з них. 2. Оцініть масштаби проекту, з різноманітними джерелами інформації.	Пом’якшення	Перегляд оцінок масштабів проекту. Переорієнтація стратегії виконання проекту.

Продовження таблиці Б.5

RS_10	Відкритий	Часті модифікації в технічному завданні	Середня	Середній	4	1.Визначити всі критичні параметри проекту 2.Чітко сформулювати вимоги до проекту 3.Проконсультуватись щодо технічних засобів та умов реалізації проекту	Перенос	Досягнути консенсусу з замовником, за потреби, внести коригування.
RS_11	Відкритий	Брак резервних копій даних	Низька	Середній	2	1. Налаштувати автоматичне архівування даних. 2. Використовувати різні носії для зберігання даних.	Попередження	Створювати резервні копії даних після завершення кожного етапу.
RS_12	Відкритий	Використання неоптимальної технології для розробки	Середня	Високий	6	1. Дослідити доступні методики і технології для реалізації проекту. 2. Вибрати найбільш підходящу та просту у застосуванні технологію.	Пом’якшення	Виділити ресурси на вдосконалення вибраної технології. Використовувати допоміжні засоби.

Продовження таблиці Б.5

RS_13	Відкритий	Помилки в плануванні	Висока	Високий	9	Тісно співпрацювати з замовником на етапі планування та регулярно демонструвати поточний прогрес.	Пом’якшення	Проводити проміжний моніторинг досягнутих результатів протягом виконання проекту.
RS_14	Відкритий	Проблеми в роботі програмного забезпечення	Низька	Високий	3	1. Організувати резервну версію програмного забезпечення. 2. Викликати фахівця для виправлення помилок.	Попередження	Замінити поточну версію програмного забезпечення.
RS_15	Відкритий	Відсутність ефективного моніторингу проекту	Середня	Низький	2	1. Регулярно контролювати прогрес проекту. 2. Забезпечити моніторинг виконання проекту з боку команди	Перенос	Здійснювати нагляд за проектом замовником. Представляти звіти про виконання проекту на кожному етапі.
RS_16	Відкритий	Втілення непотрібних функцій	Низька	Низький	1	Інформувати замовника про можливість включення додаткових функцій.	Використання	Обговорити переваги та недоліки можливих змін у проекті.

ДОДАТОК В

ЛІСТИНГИ ОСНОВНИХ ПРОГРАМНИХ МОДУЛІВ

DBControl.h

```
#pragma once
#include "DataBase.h"
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
namespace CppWinForms {
	/// <summary>
	/// Summary for DBControl
	/// </summary>
	public ref class DBControl : public
System::Windows::Forms::UserControl
	{
	public:
		DBControl(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
			DataBase db;
			dataGridView->DataSource = db.GetData()-
>Tables["thermodynamics"];
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~DBControl()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::DataGridView^ dataGridView;
	protected:
```

```
     protected:
     private:
          /// <summary>
          /// Required designer variable.
          /// </summary>
          System::ComponentModel::Container ^components;
#pragma region Windows Form Designer generated code
          /// <summary>
          /// Required method for Designer support - do not modify
          /// the contents of this method with the code editor.
          /// </summary>
          void InitializeComponent(void)
          {
               this->dataGridView                =                (gcnew
System::Windows::Forms::DataGridView());

     (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this
->dataGridView))->BeginInit();
               this->SuspendLayout();
               //
               // dataGridView
               //
               this->dataGridView->BackgroundColor                    =
System::Drawing::Color::White;
               this->dataGridView->ColumnHeadersHeightSizeMode        =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoS
ize;
               this->dataGridView->Dock                               =
System::Windows::Forms::DockStyle::Fill;
               this->dataGridView->Location = System::Drawing::Point(0,
0);
               this->dataGridView->Name = L"dataGridView";
               this->dataGridView->RowHeadersWidth = 51;
               this->dataGridView->RowTemplate->Height = 24;
               this->dataGridView->Size   =   System::Drawing::Size(524,
300);
               this->dataGridView->TabIndex = 0;
               //
               // DBControl
               //
               this->AutoScaleDimensions   =   System::Drawing::SizeF(8,
16);
               this->AutoScaleMode                                    =
System::Windows::Forms::AutoScaleMode::Font;
               this->Controls->Add(this->dataGridView);
               this->Name = L"DBControl";
               this->Size = System::Drawing::Size(524, 300);

     (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this
->dataGridView))->EndInit();
               this->ResumeLayout(false);
          }
```

```
#pragma endregion
     };
}
```

DBControl.cpp

```
#include "DBControl.h"
```

DataBase.h

```
#pragma once
/*#include <Windows.h>
#include <vcclr.h>*/

//#include <windows.h>
using namespace System::Data;
using namespace System::Data::SqlClient;
using namespace System::Windows::Forms;

ref class DataBase
{
public:
    DataBase();
    ~DataBase();
    bool Connect();
    bool Insert(double massFlowRate, double velocity, double pressure,
double      temperature,      double      adiabaticEnthalpy,      double
isochoricPressure, double adiabaticTemperature);
    bool   Insert_opt(double   massFlowRate,   double   radius,   double
plosh_pererizy);
    DataSet^ GetData();
private:
    SqlConnection^ connection;
};
```

DataBase.cpp

```
#include "DataBase.h"

DataBase::DataBase()
{
    try
    {
        connection   =   gcnew   SqlConnection("Data   Source=localhost;
Initial Catalog=therm; User ID=root; Password=MariaNed@123;");
```

```
        connection->Open();
    }
    catch (SqlException^ ex)
    {

        MessageBox::Show("Error connecting to the database : " + ex-
>Message);
    }
}

DataBase::~DataBase()
{
    connection->Close();
    delete connection;
}

bool DataBase::Connect()
{
    return (connection != nullptr && connection->State ==
ConnectionState::Open);
}

bool DataBase::Insert(double massFlowRate, double velocity, double
pressure, double temperature,
    double adiabaticEnthalpy, double isochoricPressure, double
adiabaticTemperature)
{
    try
    {

        SqlCommand^ command = gcnew SqlCommand("INSERT INTO
thermodynamics (mass_flow_rate, velocity, pressure, temperature, "
            "adiabatic_enthalpy, isochoric_pressure,
adiabatic_temperature) "
            "VALUES (@massFlowRate, @velocity, @pressure,
@temperature, "
            "@adiabaticEnthalpy, @isochoricPressure,
@adiabaticTemperature)", connection);

        // Add parameters to the query
        command->Parameters->AddWithValue("@massFlowRate",
massFlowRate);
        command->Parameters->AddWithValue("@velocity", velocity);
        command->Parameters->AddWithValue("@pressure", pressure);
        command->Parameters->AddWithValue("@temperature",
temperature);
        command->Parameters->AddWithValue("@adiabaticEnthalpy",
adiabaticEnthalpy);
        command->Parameters->AddWithValue("@isochoricPressure",
isochoricPressure);
        command->Parameters->AddWithValue("@adiabaticTemperature",
adiabaticTemperature);
```

```
        // Execute the INSERT query
        int rowsAffected = command->ExecuteNonQuery();

        return (rowsAffected > 0); // Return true if at least one row
was affected
    }
    catch (SqlException^ ex)
    {
        MessageBox::Show("Error inserting data: " + ex->Message);
        // Add any other necessary actions in case of an error
        return false;
    }
}

bool DataBase::Insert_opt(double massFlowRate, double radius, double
plosh_pererizy)
{
    try
    {

        SqlCommand^ command = gcnew SqlCommand("INSERT INTO optimalway
(mass_flow_rate, radius, plosh_pererizy) "
            "VALUES    (@massFlowRate,    @radius,    @plosh_pererizy)",
connection);

        // Add parameters to the query
        command->Parameters->AddWithValue("@massFlowRate",
massFlowRate);
        command->Parameters->AddWithValue("@radius", radius);
        command->Parameters->AddWithValue("@plosh_pererizy",
plosh_pererizy);

        // Execute the INSERT query
        int rowsAffected = command->ExecuteNonQuery();

        return (rowsAffected > 0); // Return true if at least one row
was affected
    }
    catch (SqlException^ ex)
    {
        MessageBox::Show("Error inserting data: " + ex->Message);
        // Add any other necessary actions in case of an error
        return false;
    }
}


DataSet^ DataBase::GetData()
{
```

```
    SqlCommand^ command = gcnew SqlCommand("SELECT * FROM
thermodynamics", connection);

    SqlDataAdapter^ adapter = gcnew SqlDataAdapter(command);
    DataSet^ dataSet = gcnew DataSet();
    adapter->Fill(dataSet, "your_table");

    return dataSet;
}
```

Thermodynamics.h

```
#pragma once
#include <cmath>


const double pi = 3.14159265358979323846;


class Thermodynamics {
public:
    Thermodynamics();
    void setParameters(double density, double specific_heat, double
initial_pressure, double initial_temperature, double radius, double
area);
    void calculate();
    double getMassFlowRate() const;
    double getVelocity() const;
    double getPressure() const;
    double getTemperature() const;
    double getAdiabaticEnthalpy() const;
    double getIsochoricPressure() const;
    double getAdiabaticTemperature() const;

private:

    double num_p_;
    double density_;
    double specific_heat_;
    double pressure_initial_;
    double adiabatic_enthalpy_;
    double isentropic_enthalpy_;
    double temperature_initial_;

    double radius_;
    double area_;
    double mass_flow_rate_;
    double velocity_;
    double pressure_;
```

```
    double temperature_;
    double isentropic_efficiency_;
    double isochoric_pressure_;
    double adiabatic_temperature_;

    double mass_flow();
    double velocity();
    double pressure();
    double temperature();
    double adiabatic_enthalpy();
    double isentropic_efficiency();
    double isochoric_pressure();

    double adiabatic_temperature();
};
```

Thermodynamics.cpp

```
#include "Thermodynamics.h"


Thermodynamics::Thermodynamics()
{
    num_p_ = 0;
    density_ = 0;
    specific_heat_ = 0;
    pressure_initial_ = 0;
    temperature_initial_ = 0;
    radius_ = 0;
    area_ = 0;
}

void Thermodynamics::setParameters(double density, double
specific_heat, double initial_pressure, double initial_temperature,
double radius, double area) {
    density_ = density;
    specific_heat_ = specific_heat;
    pressure_initial_ = initial_pressure;
    temperature_initial_ = initial_temperature;
    radius_ = radius;
    area_ = area;
}

void Thermodynamics::calculate() {
    mass_flow_rate_ = mass_flow();
    velocity_ = velocity();
    pressure_ = pressure();
    temperature_ = temperature();
    adiabatic_enthalpy_ = adiabatic_enthalpy();
    isentropic_efficiency_ = isentropic_efficiency();
```

```
    isochoric_pressure_ = isochoric_pressure();
    adiabatic_temperature_ = adiabatic_temperature();

}

double Thermodynamics::getMassFlowRate() const {
    return mass_flow_rate_;
}

double Thermodynamics::getVelocity() const {
    return velocity_;
}

double Thermodynamics::getPressure() const {
    return pressure_;
}

double Thermodynamics::getTemperature() const {
    return temperature_;
}

double Thermodynamics::getAdiabaticEnthalpy() const {
    return adiabatic_enthalpy_;
}

double Thermodynamics::getIsochoricPressure() const {
    return isochoric_pressure_;
}

double Thermodynamics::getAdiabaticTemperature() const {
    return adiabatic_temperature_;
}


double Thermodynamics::mass_flow() {
    return density_ * velocity() * pi * radius_ * radius_;
}

double Thermodynamics::velocity() {
    return 0.1 / (density_ * area_);
}

double Thermodynamics::pressure() {
    return 0.5 * density_ * velocity_ * velocity_ * pi * radius_ *
radius_;
}

double Thermodynamics::temperature() {
    return pressure_ / (density_ * specific_heat_);
}

double Thermodynamics::adiabatic_enthalpy() {
```

```
    return specific_heat_ * temperature_ * (pressure_ /
pressure_initial_);
}


double Thermodynamics::isentropic_efficiency() {
    return adiabatic_enthalpy_ / isentropic_enthalpy_;
}

double Thermodynamics::isochoric_pressure() {
    return pressure_initial_ * (temperature_ / temperature_initial_);
}

double Thermodynamics::adiabatic_temperature() {
    return temperature_initial_ * (pressure_ / pressure_initial_);
}
```

InputControl.h

```
#pragma once

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace std;


namespace CppWinForms {

     /// <summary>
     /// Summary for InputControl
     /// </summary>
     public ref class InputControl : public
System::Windows::Forms::UserControl
     {
     public:
          InputControl(void)
          {
               InitializeComponent();
               InitializeToolTips();
               //
               //TODO: Add the constructor code here
               //
          }

     protected:
          /// <summary>
```

```
        /// Clean up any resources being used.
        /// </summary>
        ~InputControl()
        {
             if (components)
             {
                  delete components;
             }
        }
    private: System::Windows::Forms::Label^ label1;
    protected:
    private: System::Windows::Forms::Label^ label3;
    private: System::Windows::Forms::Label^ label6;
    private: System::Windows::Forms::TextBox^ textBox_3;
    private: System::Windows::Forms::TextBox^ textBox_4;
    private: System::Windows::Forms::Label^ label7;
    private: System::Windows::Forms::Label^ label10;
    private: System::Windows::Forms::Label^ label11;
    private: System::Windows::Forms::Panel^ panel1;
    private: System::Windows::Forms::Panel^ panel2;
    private: System::Windows::Forms::Panel^ panel3;
    private: System::Windows::Forms::Panel^ panel5;
    private: System::Windows::Forms::TextBox^ textBox_1;
    private: System::Windows::Forms::TextBox^ textBox5;
    private: System::Windows::Forms::Label^ label2;
    private: System::Windows::Forms::Label^ label4;
    private: System::Windows::Forms::Panel^ panel4;
    private: System::Windows::Forms::ToolTip^ toolTip;
    public:

        String^ GetText1();
        String^ GetText3();
        String^ GetText4();
        String^ GetText5();
        bool CheckField();
    private:
        /// <summary>
        /// Required designer variable.
        /// </summary>
        System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>

        void InitializeComponent(void)
        {
             this->label1 = (gcnew System::Windows::Forms::Label());
             this->label3 = (gcnew System::Windows::Forms::Label());
             this->label6 = (gcnew System::Windows::Forms::Label());
```

```
            this->textBox_1                 =                 (gcnew
System::Windows::Forms::TextBox());
            this->textBox_3                 =                 (gcnew
System::Windows::Forms::TextBox());
            this->textBox_4                 =                 (gcnew
System::Windows::Forms::TextBox());
            this->label7 = (gcnew System::Windows::Forms::Label());
            this->label10 = (gcnew System::Windows::Forms::Label());
            this->label11 = (gcnew System::Windows::Forms::Label());
            this->panel1 = (gcnew System::Windows::Forms::Panel());
            this->panel2 = (gcnew System::Windows::Forms::Panel());
            this->panel3 = (gcnew System::Windows::Forms::Panel());
            this->panel4 = (gcnew System::Windows::Forms::Panel());
            this->panel5 = (gcnew System::Windows::Forms::Panel());
            this->textBox5                  =                 (gcnew
System::Windows::Forms::TextBox());
            this->label2 = (gcnew System::Windows::Forms::Label());
            this->label4 = (gcnew System::Windows::Forms::Label());
            this->SuspendLayout();
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->Font              =              (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label1->Location = System::Drawing::Point(68, 87);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(198, 18);
            this->label1->TabIndex = 0;
            this->label1->Text = L"Введіть кількість вузлів";
            //
            // label3
            //
            this->label3->AutoSize = true;
            this->label3->Font              =              (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label3->Location   =   System::Drawing::Point(68,
251);
            this->label3->Name = L"label3";
            this->label3->Size = System::Drawing::Size(180, 18);
            this->label3->TabIndex = 2;
            this->label3->Text = L"Початковий тиск газу";
            //
            // label6
            //
            this->label6->AutoSize = true;
```

```
            this->label6->Font                =                (gcnew
System::Drawing::Font(L"Microsoft      Sans      Serif",      8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
            this->label6->Location    =    System::Drawing::Point(31,
218);
            this->label6->Name = L"label6";
            this->label6->Size = System::Drawing::Size(235, 18);
            this->label6->TabIndex = 3;
            this->label6->Text = L"Початкова температура газу";
            //
            // textBox_1
            //
            this->textBox_1->BorderStyle                              =
System::Windows::Forms::BorderStyle::None;
            this->textBox_1->Font              =                (gcnew
System::Drawing::Font(L"Microsoft       Sans       Serif",       7.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
            this->textBox_1->Location  =  System::Drawing::Point(315,
90);
            this->textBox_1->Name = L"textBox_1";
            this->textBox_1->Size = System::Drawing::Size(135, 15);
            this->textBox_1->TabIndex = 6;
            //
            // textBox_3
            //
            this->textBox_3->BorderStyle                              =
System::Windows::Forms::BorderStyle::None;
            this->textBox_3->Font              =                (gcnew
System::Drawing::Font(L"Microsoft       Sans       Serif",       7.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
            this->textBox_3->Location  =  System::Drawing::Point(313,
249);
            this->textBox_3->Name = L"textBox_3";
            this->textBox_3->Size = System::Drawing::Size(135, 15);
            this->textBox_3->TabIndex = 8;
            //
            // textBox_4
            //
            this->textBox_4->BorderStyle                              =
System::Windows::Forms::BorderStyle::None;
            this->textBox_4->Font              =                (gcnew
System::Drawing::Font(L"Microsoft       Sans       Serif",       7.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
```

```
            this->textBox_4->Location = System::Drawing::Point(314,
218);
            this->textBox_4->Name = L"textBox_4";
            this->textBox_4->Size = System::Drawing::Size(135, 15);
            this->textBox_4->TabIndex = 9;
            //
            // label7
            //
            this->label7->AutoSize = true;
            this->label7->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 7.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label7->Location = System::Drawing::Point(107,
11);
            this->label7->Name = L"label7";
            this->label7->Size = System::Drawing::Size(364, 32);
            this->label7->TabIndex = 12;
            this->label7->Text = L"Ввід інформації для обчислення
термодинаміки \r\n та проектування шляху тру"
                L"бопроводу ";
            this->label7->Click += gcnew System::EventHandler(this,
&InputControl::label7_Click);
            //
            // label10
            //
            this->label10->AutoSize = true;
            this->label10->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label10->Location = System::Drawing::Point(463,
249);
            this->label10->Name = L"label10";
            this->label10->Size = System::Drawing::Size(29, 18);
            this->label10->TabIndex = 15;
            this->label10->Text = L"Па";
            //
            // label11
            //
            this->label11->AutoSize = true;
            this->label11->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label11->Location = System::Drawing::Point(464,
220);
            this->label11->Name = L"label11";
            this->label11->Size = System::Drawing::Size(19, 18);
```

```
            this->label11->TabIndex = 16;
            this->label11->Text = L"K";
            //
            // panel1
            //
            this->panel1->BackColor = System::Drawing::Color::Gray;
            this->panel1->Location   =   System::Drawing::Point(315,
109);
            this->panel1->Name = L"panel1";
            this->panel1->Size = System::Drawing::Size(143, 3);
            this->panel1->TabIndex = 19;
            //
            // panel2
            //
            this->panel2->BackColor = System::Drawing::Color::Gray;
            this->panel2->Location    =    System::Drawing::Point(27,
185);
            this->panel2->Name = L"panel2";
            this->panel2->Size = System::Drawing::Size(465, 2);
            this->panel2->TabIndex = 20;
            //
            // panel3
            //
            this->panel3->BackColor = System::Drawing::Color::Gray;
            this->panel3->Location   =   System::Drawing::Point(314,
266);
            this->panel3->Name = L"panel3";
            this->panel3->Size = System::Drawing::Size(143, 3);
            this->panel3->TabIndex = 20;
            //
            // panel4
            //
            this->panel4->BackColor = System::Drawing::Color::Gray;
            this->panel4->Location   =   System::Drawing::Point(315,
235);
            this->panel4->Name = L"panel4";
            this->panel4->Size = System::Drawing::Size(143, 3);
            this->panel4->TabIndex = 20;
            //
            // panel5
            //
            this->panel5->BackColor = System::Drawing::Color::Gray;
            this->panel5->Location   =   System::Drawing::Point(314,
160);
            this->panel5->Name = L"panel5";
            this->panel5->Size = System::Drawing::Size(143, 3);
            this->panel5->TabIndex = 23;
            //
            // textBox5
            //
            this->textBox5->BorderStyle                              =
System::Windows::Forms::BorderStyle::None;
```

```
            this->textBox5->Font                =                (gcnew
System::Drawing::Font(L"Microsoft       Sans       Serif",       7.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
            this->textBox5->Location  =  System::Drawing::Point(314,
141);
            this->textBox5->Name = L"textBox5";
            this->textBox5->Size = System::Drawing::Size(135, 15);
            this->textBox5->TabIndex = 22;
            this->textBox5->TextChanged           +=           gcnew
System::EventHandler(this, &InputControl::textBox5_TextChanged);
            //
            // label2
            //
            this->label2->AutoSize = true;
            this->label2->Font                =                (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
            this->label2->Location    =    System::Drawing::Point(68,
139);
            this->label2->Name = L"label2";
            this->label2->Size = System::Drawing::Size(180, 18);
            this->label2->TabIndex = 21;
            this->label2->Text = L"Об'єм газу для шляху";
            //
            // label4
            //
            this->label4->AutoSize = true;
            this->label4->Font                =                (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
            this->label4->Location   =   System::Drawing::Point(474,
146);
            this->label4->Name = L"label4";
            this->label4->Size = System::Drawing::Size(29, 18);
            this->label4->TabIndex = 24;
            this->label4->Text = L"м3";
            this->label4->Click += gcnew System::EventHandler(this,
&InputControl::label4_Click);
            //
            // InputControl
            //
            this->AutoScaleDimensions   =   System::Drawing::SizeF(8,
16);
            this->AutoScaleMode                                    =
System::Windows::Forms::AutoScaleMode::Font;
            this->BackColor = System::Drawing::Color::White;
```

```
                this->Controls->Add(this->label4);
                this->Controls->Add(this->panel5);
                this->Controls->Add(this->textBox_1);
                this->Controls->Add(this->label2);
                this->Controls->Add(this->panel4);
                this->Controls->Add(this->panel3);
                this->Controls->Add(this->panel2);
                this->Controls->Add(this->panel1);
                this->Controls->Add(this->label11);
                this->Controls->Add(this->label10);
                this->Controls->Add(this->label7);
                this->Controls->Add(this->textBox_4);
                this->Controls->Add(this->textBox_3);
                this->Controls->Add(this->textBox5);
                this->Controls->Add(this->label6);
                this->Controls->Add(this->label3);
                this->Controls->Add(this->label1);
                this->Name = L"InputControl";
                this->Size = System::Drawing::Size(524, 300);
                this->Load    +=    gcnew    System::EventHandler(this,
&InputControl::InputControl_Load);
                this->ResumeLayout(false);
                this->PerformLayout();

           }
#pragma endregion
           void InitializeToolTips()
           {
                toolTip = gcnew System::Windows::Forms::ToolTip();
                toolTip->AutoPopDelay = 5000;
                toolTip->InitialDelay = 1000;
                toolTip->ReshowDelay = 500;
                toolTip->ShowAlways = true;

                toolTip->SetToolTip(this->textBox_1, "Значення має бути
цілим числом, від 3 до 5");
                toolTip->SetToolTip(this->textBox_4,  "Значення  бажано
щоб було цілим, та в діапазоні від -40 до 85");
                toolTip->SetToolTip(this->textBox5, "Значення бажано щоб
було цілим та більше 1, або з комою роздільником");
                toolTip->SetToolTip(this->textBox_3, "Введіть значення у
діапазоні від 1200 до 7 000 000");
           }
     private:    System::Void    label7_Click(System::Object^    sender,
System::EventArgs^ e) {
     }
private:    System::Void    InputControl_Load(System::Object^    sender,
System::EventArgs^ e) {
}
private:     System::Void     label4_Click(System::Object^     sender,
System::EventArgs^ e) {
}
```

```
private:   System::Void   textBox5_TextChanged(System::Object^   sender,
System::EventArgs^ e) {
}
};
}
```

InputControl.cpp

```
#include "InputControl.h"


String^ CppWinForms::InputControl::GetText1()
{
     return textBox_1->Text;
}

String^ CppWinForms::InputControl::GetText5()
{
     return textBox5->Text;
}

String^ CppWinForms::InputControl::GetText4()
{
     return textBox_4->Text;
}

String^ CppWinForms::InputControl::GetText3()
{
     return textBox_3->Text;
}

bool CppWinForms::InputControl::CheckField()
{
     if              (String::IsNullOrEmpty(GetText1())              &&
String::IsNullOrEmpty(GetText3()) &&
          String::IsNullOrEmpty(GetText4())                          &&
String::IsNullOrEmpty(GetText5())) {
          return false;
     }

     return true;
}
```

OutputControl.h

```
#pragma once
#include "Thermodynamics.h"
```

```
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;


namespace CppWinForms {

     /// <summary>
     /// Summary for OutputControl
     /// </summary>
     public      ref      class      OutputControl      :      public
System::Windows::Forms::UserControl
     {
     public:
          OutputControl(void)
          {
               InitializeComponent();
               //
               //TODO: Add the constructor code here
               //
          }


          void DispalyData(double value1, double value2, double value3)
{
               if (isnan(value1) || isnan(value2) || isnan(value3)) {
                    Label_1->Text = "N/A";
                    Label_2->Text = "N/A";
                    Label_3->Text = "N/A";
               }
               else {
                    Label_1->Text = value1.ToString();
                    Label_2->Text = value2.ToString();
                    Label_3->Text = value3.ToString();
               }
          }


     protected:
          /// <summary>
          /// Clean up any resources being used.
          /// </summary>
          ~OutputControl()
          {
               if (components)
               {
                    delete components;
               }
```

```
        }

    protected:



    private: System::Windows::Forms::Label^ label3;
    private: System::Windows::Forms::Label^ label2;
    private: System::Windows::Forms::Label^ label1;




    private: System::Windows::Forms::Label^ Label_4;



    private: System::Windows::Forms::Label^ label7;
    private: System::Windows::Forms::Panel^ panel1;
    private: System::Windows::Forms::Panel^ panel2;
    private: System::Windows::Forms::Panel^ panel3;
    private: System::Windows::Forms::Panel^ panel5;
    private: System::Windows::Forms::Label^ label4;
    private: System::Windows::Forms::Label^ label6;
    private: System::Windows::Forms::Label^ label8;
    private: System::Windows::Forms::Label^ label5;
    private: System::Windows::Forms::Label^ label9;
    private: System::Windows::Forms::Label^ label10;
    private: System::Windows::Forms::Label^ label11;
    private: System::Windows::Forms::Label^ label12;
    private: System::Windows::Forms::Label^ label13;
    private: System::Windows::Forms::Label^ label14;
    private: System::Windows::Forms::Label^ label15;
    private: System::Windows::Forms::Panel^ panel4;




    public:

        String^ GetText1();
        String^ GetText2();
        String^ GetText3();
        bool CheckField();

    private:
        /// <summary>
        /// Required designer variable.
        /// </summary>
        System::ComponentModel::Container ^components;
#pragma region Windows Form Designer generated code
```

```
        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        void InitializeComponent(void)
        {
             this->label3 = (gcnew System::Windows::Forms::Label());
             this->label2 = (gcnew System::Windows::Forms::Label());
             this->label1 = (gcnew System::Windows::Forms::Label());
             this->Label_4 = (gcnew System::Windows::Forms::Label());
             this->label7 = (gcnew System::Windows::Forms::Label());
             this->panel1 = (gcnew System::Windows::Forms::Panel());
             this->panel2 = (gcnew System::Windows::Forms::Panel());
             this->panel3 = (gcnew System::Windows::Forms::Panel());
             this->panel4 = (gcnew System::Windows::Forms::Panel());
             this->panel5 = (gcnew System::Windows::Forms::Panel());
             this->label4 = (gcnew System::Windows::Forms::Label());
             this->label6 = (gcnew System::Windows::Forms::Label());
             this->label8 = (gcnew System::Windows::Forms::Label());
             this->label5 = (gcnew System::Windows::Forms::Label());
             this->label9 = (gcnew System::Windows::Forms::Label());
             this->label10 = (gcnew System::Windows::Forms::Label());
             this->label11 = (gcnew System::Windows::Forms::Label());
             this->label12 = (gcnew System::Windows::Forms::Label());
             this->label13 = (gcnew System::Windows::Forms::Label());
             this->label14 = (gcnew System::Windows::Forms::Label());
             this->label15 = (gcnew System::Windows::Forms::Label());
             this->SuspendLayout();
             //
             // label3
             //
             this->label3->AutoSize = true;
             this->label3->Font                =                (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
             this->label3->Location    =    System::Drawing::Point(22,
233);
             this->label3->Name = L"label3";
             this->label3->Size = System::Drawing::Size(108, 18);
             this->label3->TabIndex = 15;
             this->label3->Text = L"Радіус труби";
             //
             // label2
             //
             this->label2->AutoSize = true;
             this->label2->Font                =                (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
```

```
            this->label2->Location    =    System::Drawing::Point(35,
156);
            this->label2->Name = L"label2";
            this->label2->Size = System::Drawing::Size(79, 18);
            this->label2->TabIndex = 14;
            this->label2->Text = L"Пр. потік";
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->Font                =                (gcnew
System::Drawing::Font(L"Microsoft      Sans      Serif",      8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
            this->label1->Location = System::Drawing::Point(43, 79);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(71, 18);
            this->label1->TabIndex = 13;
            this->label1->Text = L"Ділянка";
            //
            // Label_4
            //
            this->Label_4->AutoSize = true;
            this->Label_4->Font                =                (gcnew
System::Drawing::Font(L"Microsoft      Sans      Serif",      8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
            this->Label_4->Location   =   System::Drawing::Point(348,
79);
            this->Label_4->Name = L"Label_4";
            this->Label_4->Size = System::Drawing::Size(0, 18);
            this->Label_4->TabIndex = 25;
            //
            // label7
            //
            this->label7->AutoSize = true;
            this->label7->Font                =                (gcnew
System::Drawing::Font(L"Microsoft       Sans       Serif",       7.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
            this->label7->Location   =   System::Drawing::Point(135,
25);
            this->label7->Name = L"label7";
            this->label7->Size = System::Drawing::Size(357, 16);
            this->label7->TabIndex = 19;
            this->label7->Text  =  L"Оптимальний  шлях  прокладання
трубопроводу";
            //
            // panel1
```

```
            //
            this->panel1->BackColor = System::Drawing::Color::Gray;
            this->panel1->Location = System::Drawing::Point(38, 52);
            this->panel1->Name = L"panel1";
            this->panel1->Size = System::Drawing::Size(529, 2);
            this->panel1->TabIndex = 29;
            //
            // panel2
            //
            this->panel2->BackColor = System::Drawing::Color::Gray;
            this->panel2->Location   =   System::Drawing::Point(43,
126);
            this->panel2->Name = L"panel2";
            this->panel2->Size = System::Drawing::Size(524, 2);
            this->panel2->TabIndex = 30;
            //
            // panel3
            //
            this->panel3->BackColor = System::Drawing::Color::Gray;
            this->panel3->Location   =   System::Drawing::Point(43,
201);
            this->panel3->Name = L"panel3";
            this->panel3->Size = System::Drawing::Size(528, 2);
            this->panel3->TabIndex = 30;
            //
            // panel4
            //
            this->panel4->BackColor = System::Drawing::Color::Gray;
            this->panel4->Location   =   System::Drawing::Point(43,
272);
            this->panel4->Name = L"panel4";
            this->panel4->Size = System::Drawing::Size(526, 2);
            this->panel4->TabIndex = 30;
            //
            // panel5
            //
            this->panel5->BackColor = System::Drawing::Color::Gray;
            this->panel5->Location   =   System::Drawing::Point(138,
52);
            this->panel5->Name = L"panel5";
            this->panel5->Size = System::Drawing::Size(2, 222);
            this->panel5->TabIndex = 31;
            //
            // label4
            //
            this->label4->AutoSize = true;
            this->label4->Font               =               (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
```

```
            this->label4->Location   =   System::Drawing::Point(171,
79);
            this->label4->Name = L"label4";
            this->label4->Size = System::Drawing::Size(42, 18);
            this->label4->TabIndex = 32;
            this->label4->Text = L"1 - 2";
            //
            // label6
            //
            this->label6->AutoSize = true;
            this->label6->Font                =                (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label6->Location   =   System::Drawing::Point(319,
79);
            this->label6->Name = L"label6";
            this->label6->Size = System::Drawing::Size(42, 18);
            this->label6->TabIndex = 33;
            this->label6->Text = L"2 - 3";
            //
            // label8
            //
            this->label8->AutoSize = true;
            this->label8->Font                =                (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label8->Location   =   System::Drawing::Point(452,
79);
            this->label8->Name = L"label8";
            this->label8->Size = System::Drawing::Size(42, 18);
            this->label8->TabIndex = 34;
            this->label8->Text = L"3 - 5";
            //
            // label5
            //
            this->label5->AutoSize = true;
            this->label5->Font                =                (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label5->Location   =   System::Drawing::Point(452,
233);
            this->label5->Name = L"label5";
            this->label5->Size = System::Drawing::Size(26, 18);
            this->label5->TabIndex = 38;
            this->label5->Text = L"10";
            //
```

```
            // label9
            //
            this->label9->AutoSize = true;
            this->label9->Font                =                (gcnew
System::Drawing::Font(L"Microsoft      Sans      Serif",      8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label9->Location   =   System::Drawing::Point(319,
233);
            this->label9->Name = L"label9";
            this->label9->Size = System::Drawing::Size(26, 18);
            this->label9->TabIndex = 37;
            this->label9->Text = L"15";
            //
            // label10
            //
            this->label10->AutoSize = true;
            this->label10->Font               =                (gcnew
System::Drawing::Font(L"Microsoft      Sans      Serif",      8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label10->Location   =   System::Drawing::Point(171,
233);
            this->label10->Name = L"label10";
            this->label10->Size = System::Drawing::Size(26, 18);
            this->label10->TabIndex = 36;
            this->label10->Text = L"20";
            //
            // label11
            //
            this->label11->AutoSize = true;
            this->label11->Font               =                (gcnew
System::Drawing::Font(L"Microsoft      Sans      Serif",      8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label11->Location   =   System::Drawing::Point(348,
233);
            this->label11->Name = L"label11";
            this->label11->Size = System::Drawing::Size(0, 18);
            this->label11->TabIndex = 35;
            //
            // label12
            //
            this->label12->AutoSize = true;
            this->label12->Font               =                (gcnew
System::Drawing::Font(L"Microsoft      Sans      Serif",      8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
```

```
            this->label12->Location   =   System::Drawing::Point(452,
156);
            this->label12->Name = L"label12";
            this->label12->Size = System::Drawing::Size(26, 18);
            this->label12->TabIndex = 42;
            this->label12->Text = L"12";
            //
            // label13
            //
            this->label13->AutoSize = true;
            this->label13->Font               =               (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
            this->label13->Location   =   System::Drawing::Point(319,
156);
            this->label13->Name = L"label13";
            this->label13->Size = System::Drawing::Size(17, 18);
            this->label13->TabIndex = 41;
            this->label13->Text = L"8";
            //
            // label14
            //
            this->label14->AutoSize = true;
            this->label14->Font               =               (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
            this->label14->Location   =   System::Drawing::Point(171,
156);
            this->label14->Name = L"label14";
            this->label14->Size = System::Drawing::Size(26, 18);
            this->label14->TabIndex = 40;
            this->label14->Text = L"10";
            //
            // label15
            //
            this->label15->AutoSize = true;
            this->label15->Font               =               (gcnew
System::Drawing::Font(L"Microsoft     Sans     Serif",     8.999999F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                  static_cast<System::Byte>(204)));
            this->label15->Location   =   System::Drawing::Point(348,
156);
            this->label15->Name = L"label15";
            this->label15->Size = System::Drawing::Size(0, 18);
            this->label15->TabIndex = 39;
            //
            // OutputControl
```

```
            //
            this->AutoScaleDimensions = System::Drawing::SizeF(8,
16);
            this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
            this->BackColor = System::Drawing::Color::White;
            this->Controls->Add(this->label12);
            this->Controls->Add(this->label13);
            this->Controls->Add(this->label14);
            this->Controls->Add(this->label15);
            this->Controls->Add(this->label5);
            this->Controls->Add(this->label9);
            this->Controls->Add(this->label10);
            this->Controls->Add(this->label11);
            this->Controls->Add(this->label8);
            this->Controls->Add(this->label6);
            this->Controls->Add(this->label4);
            this->Controls->Add(this->panel5);
            this->Controls->Add(this->panel4);
            this->Controls->Add(this->panel3);
            this->Controls->Add(this->panel2);
            this->Controls->Add(this->panel1);
            this->Controls->Add(this->Label_4);
            this->Controls->Add(this->label7);
            this->Controls->Add(this->label3);
            this->Controls->Add(this->label2);
            this->Controls->Add(this->label1);
            this->Name = L"OutputControl";
            this->Size = System::Drawing::Size(609, 316);
            this->Load += gcnew System::EventHandler(this,
&OutputControl::OutputControl_Load);
            this->ResumeLayout(false);
            this->PerformLayout();

        }
#pragma endregion
private: System::Void OutputControl_Load(System::Object^ sender,
System::EventArgs^ e) {
}
};
}
```

OutputControl.cpp

```
#include "OutputControl.h"

String^ CppWinForms::OutputControl::GetText1()
{
    return Label_1->Text;
}
```

```
String^ CppWinForms::OutputControl::GetText2()
{
    return Label_2->Text;
}

String^ CppWinForms::OutputControl::GetText3()
{
    return Label_3->Text;
}

bool CppWinForms::OutputControl::CheckField()
{
    if              (String::IsNullOrEmpty(GetText1())              &&
String::IsNullOrEmpty(GetText2())                                   &&
String::IsNullOrEmpty(GetText3())) {
        return false;
    }

    return true;
}
```

MyForm.h

```
#pragma once
#include "InputControl.h"
#include "InOptControl.h"
#include "OutputControl.h"
#include "DBControl.h"
#include "Thermodynamics.h"


namespace $safeprojectname$ {

     using namespace System;
     using namespace System::ComponentModel;
     using namespace System::Collections;
     using namespace System::Windows::Forms;
     using namespace System::Data;
     using namespace System::Drawing;
     using namespace CppWinForms;


     /// <summary>
     /// Summary for MyForm
     /// </summary>
     public ref class MyForm : public System::Windows::Forms::Form
     {
     public:
          MyForm(void)
          {
               InitializeComponent();
```

```
            //
            //TODO: Add the constructor code here
            //
            mode = 0;
            PlacingElement();
        }

        void PlacingElement() {

            if (mode == 0) {
                Button_Calculation->Visible = true;
                Button_Calculation2->Visible = false;
                Button_ViewDB->Visible = false;
                Button_AddToDB->Visible = false;
                icontrol = gcnew InputControl();
                icontrol->Dock                                          =
System::Windows::Forms::DockStyle::Fill;
                panelHome->Controls->Clear();
                panelHome->Controls->Add(icontrol);



            }
            else if (mode == 1) {

                Button_Calculation->Visible = false;
                Button_Calculation2->Visible = false;
                Button_ViewDB->Visible = false;
                Button_AddToDB->Visible = false;
                ocontrol = gcnew OutputControl();
                ocontrol->Dock                                          =
System::Windows::Forms::DockStyle::Fill;
                panelHome->Controls->Clear();
                panelHome->Controls->Add(ocontrol);

            }
            else if (mode == 5) {

                Button_Calculation->Visible = false;
                Button_Calculation2->Visible = true;
                Button_ViewDB->Visible = false;
                Button_AddToDB->Visible = false;
                incontrol = gcnew InOptControl();
                incontrol->Dock                                         =
System::Windows::Forms::DockStyle::Fill;
                panelHome->Controls->Clear();
                panelHome->Controls->Add(incontrol);

            }
            else {
                Button_Calculation->Visible = false;
                Button_Calculation2->Visible = false;
```

```
                    Button_ViewDB->Visible = false;
                    Button_AddToDB->Visible = false;
                    dbcontrol = gcnew DBControl();
                    dbcontrol->Dock                                    =
System::Windows::Forms::DockStyle::Fill;
                    panelHome->Controls->Clear();
                    panelHome->Controls->Add(dbcontrol);
               }
          }

     protected:
          /// <summary>
          /// Clean up any resources being used.
          /// </summary>
          ~MyForm()
          {
               if (components)
               {
                    delete thermodynamics;
                    delete components;
               }
          }
     private: System::Windows::Forms::Label^ Button_Input;

     protected:

     private: System::Windows::Forms::Label^ Button_Output;

     protected:

     private:
          /// <summary>
          /// Required designer variable.
          /// </summary>
          System::ComponentModel::Container^ components;
     private: System::Windows::Forms::Panel^ panelHome;

     private: System::Windows::Forms::Button^ Button_Calculation;
     private: System::Windows::Forms::Button^ Button_Calculation2;
     private: System::Windows::Forms::Label^ Button_Close;
     private: InputControl^ icontrol;
     private: InOptControl^ incontrol;
     private: OutputControl^ ocontrol;
     private: DBControl^ dbcontrol;
             Thermodynamics* thermodynamics = new Thermodynamics;
     private: System::Windows::Forms::Panel^ panel1;
     private: System::Windows::Forms::Panel^ panel2;

     private: System::Windows::Forms::Button^ Button_ViewDB;
     private: System::Windows::Forms::Button^ Button_AddToDB;
```

```
            int mode;

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            void InitializeComponent(void)
            {
                this->Button_Input              =              (gcnew
System::Windows::Forms::Label());
                this->Button_Output             =              (gcnew
System::Windows::Forms::Label());
                this->panelHome                 =              (gcnew
System::Windows::Forms::Panel());
                this->Button_Calculation        =              (gcnew
System::Windows::Forms::Button());
                this->Button_Calculation2       =              (gcnew
System::Windows::Forms::Button());
                this->Button_Close              =              (gcnew
System::Windows::Forms::Label());
                this->panel1                    =              (gcnew
System::Windows::Forms::Panel());
                this->panel2                    =              (gcnew
System::Windows::Forms::Panel());
                this->Button_ViewDB             =              (gcnew
System::Windows::Forms::Button());
                this->Button_AddToDB            =              (gcnew
System::Windows::Forms::Button());
                this->panel1->SuspendLayout();
                this->SuspendLayout();
                //
                // Button_Input
                //
                this->Button_Input->AutoSize = true;
                this->Button_Input->Cursor                          =
System::Windows::Forms::Cursors::Hand;
                this->Button_Input->Font        =              (gcnew
System::Drawing::Font(L"Ubuntu",                              10.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                    static_cast<System::Byte>(204)));
                this->Button_Input->ForeColor                       =
System::Drawing::Color::White;
                this->Button_Input->Location                        =
System::Drawing::Point(30, 14);
                this->Button_Input->Name = L"Button_Input";
                this->Button_Input->Size = System::Drawing::Size(185,
23);
                this->Button_Input->TabIndex = 0;
```

```
                this->Button_Input->Text = L"Ввести інформацію";
                this->Button_Input->Click          +=          gcnew
System::EventHandler(this, &MyForm::Button_Input_Click);
                this->Button_Input->MouseEnter        +=        gcnew
System::EventHandler(this, &MyForm::Button_Input_MouseEnter);
                this->Button_Input->MouseLeave        +=        gcnew
System::EventHandler(this, &MyForm::Button_Input_MouseLeave);
                //
                // Button_Output
                //
                this->Button_Output->AutoSize = true;
                this->Button_Output->Cursor                          =
System::Windows::Forms::Cursors::Hand;
                this->Button_Output->Font          =          (gcnew
System::Drawing::Font(L"Ubuntu",                                10.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                     static_cast<System::Byte>(204)));
                this->Button_Output->ForeColor                       =
System::Drawing::Color::White;
                this->Button_Output->Location                        =
System::Drawing::Point(248, 14);
                this->Button_Output->Name = L"Button_Output";
                this->Button_Output->Size                            =
System::Drawing::Size(237, 23);
                this->Button_Output->TabIndex = 1;
                this->Button_Output->Text       =       L"Переглянути
інформацію";
                this->Button_Output->Click          +=          gcnew
System::EventHandler(this, &MyForm::Button_Output_Click);
                this->Button_Output->MouseEnter       +=       gcnew
System::EventHandler(this, &MyForm::Button_Output_MouseEnter);
                this->Button_Output->MouseLeave       +=       gcnew
System::EventHandler(this, &MyForm::Button_Output_MouseLeave);
                //
                // panelHome
                //
                this->panelHome->Location                            =
System::Drawing::Point(12, 51);
                this->panelHome->Name = L"panelHome";
                this->panelHome->Size   =   System::Drawing::Size(600,
343);
                this->panelHome->TabIndex = 3;
                //
                // Button_Calculation
                //
                this->Button_Calculation->BackColor                  =
System::Drawing::Color::Black;
                this->Button_Calculation->Cursor                     =
System::Windows::Forms::Cursors::Hand;
                this->Button_Calculation->FlatStyle                  =
System::Windows::Forms::FlatStyle::Popup;
```

```
                this->Button_Calculation->Font        =        (gcnew
System::Drawing::Font(L"Ubuntu",                                10.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                     static_cast<System::Byte>(204)));
                this->Button_Calculation->ForeColor                =
System::Drawing::Color::White;
                this->Button_Calculation->Location                 =
System::Drawing::Point(-3, 400);
                this->Button_Calculation->Name                     =
L"Button_Calculation";
                this->Button_Calculation->Size                     =
System::Drawing::Size(628, 44);
                this->Button_Calculation->TabIndex = 4;
                this->Button_Calculation->Text  =  L"Почати  внесення
даних";
                this->Button_Calculation->UseVisualStyleBackColor   =
false;
                this->Button_Calculation->Click       +=       gcnew
System::EventHandler(this, &MyForm::button2_Click);
                this->Button_Calculation->MouseEnter     +=     gcnew
System::EventHandler(this, &MyForm::Button_Calculation_MouseEnter);
                this->Button_Calculation->MouseLeave     +=     gcnew
System::EventHandler(this, &MyForm::Button_Calculation_MouseLeave);
                //
   // Button_Calculation2
   //
                this->Button_Calculation2->BackColor               =
System::Drawing::Color::Black;
                this->Button_Calculation2->Cursor                  =
System::Windows::Forms::Cursors::Hand;
                this->Button_Calculation2->FlatStyle               =
System::Windows::Forms::FlatStyle::Popup;
                this->Button_Calculation2->Font        =        (gcnew
System::Drawing::Font(L"Ubuntu",                                10.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                     static_cast<System::Byte>(204)));
                this->Button_Calculation2->ForeColor               =
System::Drawing::Color::White;
                this->Button_Calculation2->Location                =
System::Drawing::Point(-3, 400);
                this->Button_Calculation2->Name                    =
L"Button_Calculation2";
                this->Button_Calculation2->Size                    =
System::Drawing::Size(628, 44);
                this->Button_Calculation2->TabIndex = 4;
                this->Button_Calculation2->Text = L"Внести дані";
                this->Button_Calculation2->UseVisualStyleBackColor  =
false;
                this->Button_Calculation2->Click       +=       gcnew
System::EventHandler(this, &MyForm::button2_Click2);
```

```
                this->Button_Calculation2->MouseEnter    +=    gcnew
System::EventHandler(this, &MyForm::Button_Calculation2_MouseEnter);
                this->Button_Calculation2->MouseLeave    +=    gcnew
System::EventHandler(this, &MyForm::Button_Calculation2_MouseLeave);
                //
                // Button_Close
                //
                this->Button_Close->AutoSize = true;
                this->Button_Close->Cursor                              =
System::Windows::Forms::Cursors::Hand;
                this->Button_Close->Font          =          (gcnew
System::Drawing::Font(L"Ubuntu",                                      10.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                     static_cast<System::Byte>(204)));
                this->Button_Close->ForeColor                           =
System::Drawing::Color::White;
                this->Button_Close->Location                            =
System::Drawing::Point(593, 14);
                this->Button_Close->Name = L"Button_Close";
                this->Button_Close->Size  =  System::Drawing::Size(22,
23);
                this->Button_Close->TabIndex = 4;
                this->Button_Close->Text = L"X";
                this->Button_Close->Click          +=          gcnew
System::EventHandler(this, &MyForm::Button_Close_Click);
                //
                // panel1
                //
                this->panel1->BackColor                                 =
System::Drawing::Color::Black;
                this->panel1->Controls->Add(this->Button_Close);
                this->panel1->Controls->Add(this->Button_Output);
                this->panel1->Controls->Add(this->Button_Input);
                this->panel1->Location = System::Drawing::Point(-3, -
5);
                this->panel1->Name = L"panel1";
                this->panel1->Size = System::Drawing::Size(628, 53);
                this->panel1->TabIndex = 5;
                //
                // panel2
                //
                this->panel2->BackColor                                 =
System::Drawing::Color::Black;
                this->panel2->Location   =   System::Drawing::Point(0,
398);
                this->panel2->Name = L"panel2";
                this->panel2->Size = System::Drawing::Size(630, 3);
                this->panel2->TabIndex = 0;
                //
                // Button_ViewDB
                //
```

```
                this->Button_ViewDB->BackColor                          =
System::Drawing::Color::Black;
                this->Button_ViewDB->Cursor                             =
System::Windows::Forms::Cursors::Hand;
                this->Button_ViewDB->FlatStyle                          =
System::Windows::Forms::FlatStyle::Popup;
                this->Button_ViewDB->Font          =          (gcnew
System::Drawing::Font(L"Ubuntu",                                  10.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                     static_cast<System::Byte>(204)));
                this->Button_ViewDB->ForeColor                          =
System::Drawing::Color::White;
                this->Button_ViewDB->Location                           =
System::Drawing::Point(0, 400);
                this->Button_ViewDB->Name = L"Button_ViewDB";
                this->Button_ViewDB->Size                               =
System::Drawing::Size(317, 44);
                this->Button_ViewDB->TabIndex = 6;
                this->Button_ViewDB->Text   =   L"Переглянути   базу
даних";
                this->Button_ViewDB->UseVisualStyleBackColor = false;
                this->Button_ViewDB->Click          +=          gcnew
System::EventHandler(this, &MyForm::Button_ViewDB_Click);
                this->Button_ViewDB->MouseEnter       +=       gcnew
System::EventHandler(this, &MyForm::Button_ViewDB_MouseEnter);
                this->Button_ViewDB->MouseLeave       +=       gcnew
System::EventHandler(this, &MyForm::Button_ViewDB_MouseLeave);
                //
                // Button_AddToDB
                //
                this->Button_AddToDB->BackColor                         =
System::Drawing::Color::Black;
                this->Button_AddToDB->Cursor                            =
System::Windows::Forms::Cursors::Hand;
                this->Button_AddToDB->FlatStyle                         =
System::Windows::Forms::FlatStyle::Popup;
                this->Button_AddToDB->Font          =          (gcnew
System::Drawing::Font(L"Ubuntu",                                  10.8F,
System::Drawing::FontStyle::Bold,
System::Drawing::GraphicsUnit::Point,
                     static_cast<System::Byte>(204)));
                this->Button_AddToDB->ForeColor                         =
System::Drawing::Color::White;
                this->Button_AddToDB->Location                          =
System::Drawing::Point(315, 400);
                this->Button_AddToDB->Name = L"Button_AddToDB";
                this->Button_AddToDB->Size                              =
System::Drawing::Size(315, 44);
                this->Button_AddToDB->TabIndex = 7;
                this->Button_AddToDB->Text   =   L"Добавити   в   базу
даних";
```

```
                this->Button_AddToDB->UseVisualStyleBackColor       =
false;
                this->Button_AddToDB->Click         +=          gcnew
System::EventHandler(this, &MyForm::Button_AddToDB_Click);
                this->Button_AddToDB->MouseEnter       +=       gcnew
System::EventHandler(this, &MyForm::Button_AddToDB_MouseEnter);
                this->Button_AddToDB->MouseLeave       +=       gcnew
System::EventHandler(this, &MyForm::Button_AddToDB_MouseLeave);
                //
                // MyForm
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(8,
16);
                this->AutoScaleMode                                 =
System::Windows::Forms::AutoScaleMode::Font;
                this->BackColor = System::Drawing::Color::White;
                this->ClientSize = System::Drawing::Size(624, 442);
                this->Controls->Add(this->Button_AddToDB);
                this->Controls->Add(this->Button_ViewDB);
                this->Controls->Add(this->panel2);
                this->Controls->Add(this->panel1);
                this->Controls->Add(this->Button_Calculation);
                this->Controls->Add(this->Button_Calculation2);
                this->Controls->Add(this->panelHome);
                this->FormBorderStyle                               =
System::Windows::Forms::FormBorderStyle::None;
                this->Name = L"MyForm";
                this->Text = L"MyForm";
                this->panel1->ResumeLayout(false);
                this->panel1->PerformLayout();
                this->ResumeLayout(false);

            }
#pragma endregion
     private:  System::Void  Button_Input_Click(System::Object^  sender,
System::EventArgs^ e) {
          mode = 0;
          PlacingElement();
     }
     private: System::Void Button_Output_Click(System::Object^ sender,
System::EventArgs^ e) {
          mode = 1;
          PlacingElement();
          //ocontrol->DispalyData(thermodynamics->getMassFlowRate());
     }
     private:   System::Void   button2_Click(System::Object^   sender,
System::EventArgs^ e) {
          if (icontrol->CheckField()) {
               double     num_p,     way_flow,     initial_temperature,
initial_pressure;
               if   (Double::TryParse(icontrol->GetText1(),   num_p)   &&
Double::TryParse(icontrol->GetText5(), way_flow)
```

```
                    &&          Double::TryParse(icontrol->GetText4(),
initial_temperature)     &&     Double::TryParse(icontrol->GetText3(),
initial_pressure)
                    && (num_p >= 3 && num_p <= 5) && way_flow >= 1 &&
(initial_temperature > -45 && initial_temperature < 85)
                    && initial_pressure >= 1200 && initial_pressure <=
7000000)
               {
                    thermodynamics->setParameters_st(num_p,   way_flow,
initial_temperature, initial_pressure);
                    mode = 5;
                    PlacingElement();
               }
               else {
                    MessageBox::Show("Не правильно введені дані!");
               }
          }
          else {
               MessageBox::Show("Не правильно введені дані!");
          }
     }

     private:   System::Void   button2_Click2(System::Object^   sender,
System::EventArgs^ e) {
          if (incontrol->CheckField()) {
               double mass_flow_rate, radius, area;
               if              (Double::TryParse(incontrol->GetText8(),
mass_flow_rate) && Double::TryParse(incontrol->GetText9(), radius)
                    &&  Double::TryParse(incontrol->GetText10(),  area)
&& mass_flow_rate >= 0
                    && (radius >= 12.5 && radius <= 610) && (area >=
0.0005 && area <= 1.17))
               {
                    thermodynamics->setParameters_opt(mass_flow_rate,
radius, area);
                    thermodynamics->calculate();

                    MessageBox::Show("Обраховано!");
               }
               else {
                    MessageBox::Show("Не правильно введені дані!");
               }
          }
          else {
               MessageBox::Show("Не правильно введені дані!");
          }
     }


private:   System::Void   Button_Close_Click(System::Object^   sender,
System::EventArgs^ e) {
          this->Close();
```

```
    }


private:  System::Void  Button_Input_MouseEnter(System::Object^  sender,
System::EventArgs^ e) {
          Button_Input->BackColor = Color::White;
          Button_Input->ForeColor = Color::Black;
    }

private:  System::Void  Button_Input_MouseLeave(System::Object^  sender,
System::EventArgs^ e) {
          Button_Input->BackColor = Color::Black;
          Button_Input->ForeColor = Color::White;
    }

private: System::Void Button_Output_MouseEnter(System::Object^ sender,
System::EventArgs^ e) {
          Button_Output->BackColor = Color::White;
          Button_Output->ForeColor = Color::Black;
    }

private: System::Void Button_Output_MouseLeave(System::Object^ sender,
System::EventArgs^ e) {
          Button_Output->BackColor = Color::Black;
          Button_Output->ForeColor = Color::White;
    }

private:   System::Void   Button_Calculation_MouseEnter(System::Object^
sender, System::EventArgs^ e) {
          Button_Calculation->BackColor = Color::White;
          Button_Calculation->ForeColor = Color::Black;
    }

private:   System::Void   Button_Calculation_MouseLeave(System::Object^
sender, System::EventArgs^ e) {
          Button_Calculation->BackColor = Color::Black;
          Button_Calculation->ForeColor = Color::White;
    }

private:  System::Void  Button_Calculation2_MouseEnter(System::Object^
sender, System::EventArgs^ e) {
          Button_Calculation->BackColor = Color::White;
          Button_Calculation->ForeColor = Color::Black;
    }

private:  System::Void  Button_Calculation2_MouseLeave(System::Object^
sender, System::EventArgs^ e) {
          Button_Calculation->BackColor = Color::Black;
          Button_Calculation->ForeColor = Color::White;
    }
```

```
private:   System::Void   Button_ViewDB_Click(System::Object^   sender,
System::EventArgs^ e) {
          mode = 2;
          PlacingElement();
     }
     private:   System::Void   Button_ViewDB_MouseEnter(System::Object^
sender, System::EventArgs^ e) {
          Button_ViewDB->BackColor = Color::White;
          Button_ViewDB->ForeColor = Color::Black;
     }
     private:   System::Void   Button_ViewDB_MouseLeave(System::Object^
sender, System::EventArgs^ e) {
          Button_ViewDB->BackColor = Color::Black;
          Button_ViewDB->ForeColor = Color::White;
     }
     private:     System::Void     Button_AddToDB_Click(System::Object^
sender, System::EventArgs^ e) {
          if (!ocontrol->CheckField())
               return;

          double massFlowRate = thermodynamics->getMassFlowRate();
          double velocity = thermodynamics->getVelocity();
          double pressure = thermodynamics->getPressure();
          double temperature = thermodynamics->getTemperature();
          double       adiabaticEnthalpy       =       thermodynamics-
>getAdiabaticEnthalpy();
          double       isochoricPressure       =       thermodynamics-
>getIsochoricPressure();
          double      adiabaticTemperature      =      thermodynamics-
>getAdiabaticTemperature();
          DataBase db;
          db.Insert(massFlowRate,   velocity,   pressure,   temperature,
adiabaticEnthalpy, isochoricPressure, adiabaticTemperature);

          MessageBox::Show("Дані добавлено в базу даних!");
     }

private:     System::Void     Button_AddToDB_MouseEnter(System::Object^
sender, System::EventArgs^ e) {
          Button_AddToDB->BackColor = Color::White;
          Button_AddToDB->ForeColor = Color::Black;
     }

private:     System::Void     Button_AddToDB_MouseLeave(System::Object^
sender, System::EventArgs^ e) {
          Button_AddToDB->BackColor = Color::Black;
          Button_AddToDB->ForeColor = Color::White;
     }
     };
}
```

MyForm.cpp

```
#include "MyForm.h"
using namespace System;
using namespace System::Windows::Forms;
[STAThreadAttribute]

void main(array<String^>^ args) {
     Application::EnableVisualStyles();
     Application::SetCompatibleTextRenderingDefault(false);
     $safeprojectname$::MyForm form;
     Application::Run(% form);
}
```