

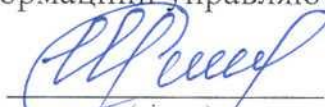
**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ**

КВАЛІФІКАЦІЙНА РОБОТА

**на тему:
«РОЗРОБКА ЗАСТОСУНКУ ДЛЯ СТВОРЕННЯ ПРОФІЛЮ ТЕПЛОВОГО
НАВАНТАЖЕННЯ БУДІВЛІ»**

Здобувача вищої освіти другого (магістерського) рівня,
II курсу, групи КМІ-23-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології в економіці»


(підпис)

Шевченко Олег Сергійович
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Горбовий Артур Юліанович,
д.т.н, професор
(прізвище, ім'я, по батькові, науковий
ступінь, вчене звання)

Керівник освітньо-
професійної програми


(підпис)

Горбовий Артур Юліанович,
д.т.н, професор
(прізвище, ім'я, по батькові, науковий
ступінь, вчене звання)

Допущено до захисту за рішенням кафедри протокол від «10» січня 2025 №11

Завідувач кафедри


(підпис)

Філоненко Михайло
Миколайович, к.ф.-м.н.,
доцент
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Ірпінь – 2025

Реєстрація № 9

27.01.2025
(дата)


(підпис відповідальної особи)

Результати перевірки

до захисту
(до захисту, на доопрацювання)

27.01.2025
(дата)


(підпис)

Результат захисту

4 (добре)
(оцінка)

06.02.2025
(дата)


(підпис)

Голова комісії:

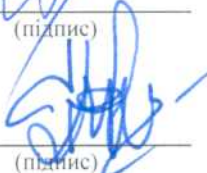

(підпис)

В.М. Франчук

Члени комісії:


(підпис)

А.Ю. Горбовий


(підпис)

М.М. Філоненко


(підпис)

І.Д. Погореловська

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст кваліфікаційної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	27		06.02.2025 Франчук В.М. Горбовий А.Ю. Філоненко М.М. Погореловська І.Д. Суховецька Л.П.
2. Дотримання вимог щодо оформлення	від 0 до 10	9		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		
4. Рецензент	від 0 до 5	4		
РАЗОМ БАЛІВ	від 0 до 50	44		

Оцінювання кваліфікаційної роботи науковим керівником - 43 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
87	B	4 (добре)	06.02.2025	Франчук В.М. Горбовий А.Ю. Філоненко М.М. Погореловська І.Д. Суховецька Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології в економіці»

Затверджую

Завідувач кафедри  Філоненко М.М.
«20» грудня 2023

**Завдання на кваліфікаційну роботу
здобувачу вищої освіти**

Шевченко Олег Сергійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «РОЗРОБКА ЗАСТОСУНКУ ДЛЯ
СТВОРЕННЯ ПРОФІЛЮ ТЕПЛООВОГО НАВАНТАЖЕННЯ БУДІВЛІ»

Затверджена наказом ректора від «20» грудня 2023 № 1701

2. Строк здачі закінченої роботи «07» січня 2025

3. Метою роботи є розробка програмного забезпечення для створення профілю теплового навантаження будівлі, яке дозволить розраховувати теплові втрати через огорожувальні конструкції, вентиляційні системи та внутрішні джерела тепла, що дозволить визначати потреби в енергії для опалення, яке необхідно для оптимізації енергетичних характеристик будівлі.

Об'єктом дослідження є процес теплопередачі в будівлях, який включає в себе теплообмін між внутрішніми і зовнішніми просторами будівлі, вплив кліматичних умов на ці процеси, а також способи зменшення теплових втрат.

Предметом дослідження є розробка програмних інструментів для оцінки теплового навантаження, що ґрунтуються на математичних моделях теплопередачі і враховують фізичні характеристики будівельних конструкцій та кліматичні умови.

4. Вихідні дані до роботи пов'язаного з розробкою застосунку для створення профілю теплового навантаження будівлі, зумовлена необхідністю удосконалення методів оцінки енергетичних характеристик будівель в умовах зміни клімату та підвищених вимог до енергоефективності.

5. Перелік графічного матеріалу відсутній

6. Зміст кваліфікаційної роботи

РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ РОЗРОБКИ
ЗАСТОСУНКУ ДЛЯ СТВОРЕННЯ ПРОФІЛЮ ТЕПЛООВОГО
НАВАНТАЖЕННЯ БУДІВЛІ

1.1. Теоретичні аспекти аналізу теплового навантаження будівель

1.1.1. Втрати теплоти крізь огорожувальні конструкції

1.1.2. Теплове навантаження на вентиляцію

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.12.2023	30.11.2023
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	27.05.2024	10.04.2024
3	II розділ (аналіз експериментальної частини про методики досліджень)	02.09.2024	22.07.2024
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	18.11.2024	04.10.2024
5	Висновки	10.12.2024	12.11.2024
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	23.12.2024	18.12.2024
7	Отримання відгуку керівника	28.01.2025	24.01.2025
8	Отримання зовнішньої рецензії	31.01.2025	28.01.2025

8. Дата видачі завдання «23» грудня 2023

9. Керівник кваліфікаційної роботи


(підпис)

Горбовий А.Ю.
(прізвище, ініціали)

10. Керівник освітньо-професійної програми


(підпис)

Горбовий А.Ю.
(прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти


(підпис)

Шевченко Олег
Сергійович
(прізвище, ініціали)

АНОТАЦІЯ ТА КЛЮЧОВІ СЛОВА

Кваліфікаційна робота, 52 с., 22 рисунки, 3 таблиці, 40 джерел.

Об'єктом розробки кваліфікаційної роботи є програмний застосунок для розрахунку теплового навантаження конструкцій будівель.

Програмний продукт створений для використання інженерами-проектувальниками, фахівцями з енергоефективності та іншими користувачами, зацікавленими в автоматизації розрахунків тепловтрат будівель. Для реалізації проекту використано мову програмування Python, бібліотеки PyQt5 для розробки інтерфейсу, а також FPDF для створення звітів.

Основний зміст роботи полягає у розробці програмного забезпечення, яке дозволяє проводити розрахунок теплового навантаження будівлі з урахуванням характеристик конструкцій, площі, внутрішньої та зовнішньої температур, а також тепловтрат від вентиляції. Результатом роботи є програмний інструмент, який забезпечує введення даних користувачем, автоматичний розрахунок параметрів та генерацію звіту у форматі PDF з деталізацією конструктивних характеристик будівлі.

Ключові слова: програмний застосунок, Python, PyQt5, FPDF, JSON, теплове навантаження, енергоефективність.

ЗМІСТ

ВСТУП	5
РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ РОЗРОБКИ ЗАСТОСУНКУ ДЛЯ СТВОРЕННЯ ПРОФІЛЮ ТЕПЛООВОГО НАВАНТАЖЕННЯ БУДІВЛІ.....	7
1.1. Теоретичні аспекти аналізу теплового навантаження будівель	7
1.1.1. Втрати теплоти крізь огорожувальні конструкції	9
1.1.2. Теплове навантаження на вентиляцію	11
1.1.3. Автоматизація розрахунків за допомогою інформаційних технологій	13
1.2. Методологічні підходи до дослідження теплового навантаження	14
1.2.1. Розрахунок теплового навантаження будівлі.....	15
1.2.2. Використання комп'ютерних технологій для аналізу теплового навантаження.....	17
РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ РОЗРОБКИ ЗАСТОСУНКУ ДЛЯ СТВОРЕННЯ ПРОФІЛЮ ТЕПЛООВОГО НАВАНТАЖЕННЯ	19
2.1. Аналіз вхідних даних для побудови профілю теплового навантаження.....	19
2.2. Розробка алгоритму побудови профілю теплового навантаження	20
2.3. Реалізація програмного забезпечення	25
2.3.1. Створення класів додатку.....	25
2.3.2. Реалізація допоміжних методів.....	28
2.3.3. Реалізація центрального файлу програми	30
2.3.4. Реалізація вкладки з інформацією про додаток	32
2.3.5. Реалізація вкладки для додавання конструкцій	33
2.3.6. Реалізація вкладки для отримання результатів розрахунків	37
РОЗДІЛ 3. ПЕРСПЕКТИВИ ВДОСКОНАЛЕНН ПІДХОДІВ ДО РОЗРАХУНКУ ТЕПЛООВОГО НАВАНТАЖЕННЯ БУДІВЕЛЬ	41
3.1. Пропозиції щодо розвитку застосунку.....	41
3.2. Рекомендації щодо впровадження розробленого застосунку та його значимість.....	43
ВИСНОВКИ.....	46
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	49
ДОДАТКИ.....	53
Додаток А	53

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

PDF	Portable Document Format
IDEF0	Метод моделювання функцій та процесів
DFD	Data Flow Diagram
UI	User Interface
UX	User Experience
JSON	JavaScript Object Notation
Вт/(м²* С⁰)	Ват на квадратний метр на цельсій

ВСТУП

Актуальність дослідження, пов'язаного з розробкою застосунку для створення профілю теплового навантаження будівлі, зумовлена необхідністю удосконалення методів оцінки енергетичних характеристик будівель в умовах зміни клімату та підвищених вимог до енергоефективності. Враховуючи значний вплив температурних коливань на енергоспоживання будівель, необхідно мати інструменти для оцінки теплових втрат, які визначають ефективність роботи систем опалення, вентиляції та кондиціонування. З урахуванням сучасних вимог до зниження енергоспоживання та підвищення комфорту в будівлях, виникає потреба у розробці спеціалізованих інструментів для розрахунку теплового навантаження, що дозволяє оптимізувати проектування енергоефективних будівель та забезпечити їх стійкість до змін клімату.

Метою цього дослідження є розробка програмного забезпечення для створення профілю теплового навантаження будівлі, яке дозволить розраховувати теплові втрати через огорожувальні конструкції, вентиляційні системи та внутрішні джерела тепла, що дозволить визначати потреби в енергії для опалення, яке необхідно для оптимізації енергетичних характеристик будівлі.

Завдання, що ставляться в рамках кваліфікаційної роботи, полягають у наступному:

- Провести аналіз існуючих методів оцінки теплових втрат і теплового навантаження будівель.
- Розробити алгоритм для обчислення теплових втрат через різні конструктивні елементи будівлі, такі як стіни, вікна, двері, дах, підлога.
- Провести аналіз вхідних даних про характеристики будівлі та кліматичні умови, їх вплив на розрахунки теплового навантаження.
- Створити інтерфейс користувача для введення даних про характеристики будівлі та кліматичні умови.

— Розробити функціонал формування звіту про теплове навантаження будівлі на основі розрахунків теплових втрат і енергетичних потреб.

— Розробити механізми для введення та обробки даних вручну, зокрема характеристик матеріалів та розмірів елементів будівлі

Об'єктом дослідження є процес теплопередачі в будівлях, який включає в себе теплообмін між внутрішніми і зовнішніми просторами будівлі, вплив кліматичних умов на ці процеси, а також способи зменшення теплових втрат.

Предметом дослідження є розробка програмних інструментів для оцінки теплового навантаження, що ґрунтуються на математичних моделях теплопередачі і враховують фізичні характеристики будівельних конструкцій та кліматичні умови.

Методи дослідження включають теоретичний аналіз існуючих методів та підходів до розрахунку теплових навантажень, математичне моделювання теплопередачі в будівлях, а також програмування для створення програмного забезпечення.

РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ РОЗРОБКИ ЗАСТОСУНКУ ДЛЯ СТВОРЕННЯ ПРОФІЛЮ ТЕПЛОВОГО НАВАНТАЖЕННЯ БУДІВЛІ

1.1. Теоретичні аспекти аналізу теплового навантаження будівель

Розрахунок теплового навантаження будівлі здійснюється для визначення характеристик опалювальних приладів та елементів системи опалення, а також для уточнення договірного навантаження, встановленого теплопостачальною організацією.

Завдання теплопостачання полягає у забезпеченні заданих умов температурного режиму в приміщеннях. Реалізація цього завдання залежить від роботи джерела теплоти, зовнішніх і внутрішніх тепломереж, а також теплоізоляції будівлі. Порушення роботи будь-якого з цих елементів може призвести до перевитрат енергоресурсів і відхилення від заданого температурного режиму [13].

Теплове навантаження будівлі визначається обсягом теплової енергії, необхідної для підтримання температурного режиму в приміщеннях. Воно охоплює тепло, що споживається системами опалення, вентиляції, гарячого водопостачання, а також іншими системами, передбаченими в будівлі [5].

Таким чином, теплове навантаження – це обсяг тепла, потрібний будівлі або приміщенню для підтримання необхідного температурного режиму протягом однієї години [16].

Теплові втрати будівлі виникають через передачу тепла через огорожувальні конструкції (стіни, покрівлю, підлогу, вікна) та через системи вентиляції. Основними факторами, які впливають на величину теплових втрат, є температура зовнішнього повітря, теплофізичні властивості матеріалів, геометричні параметри конструкцій та умови експлуатації.

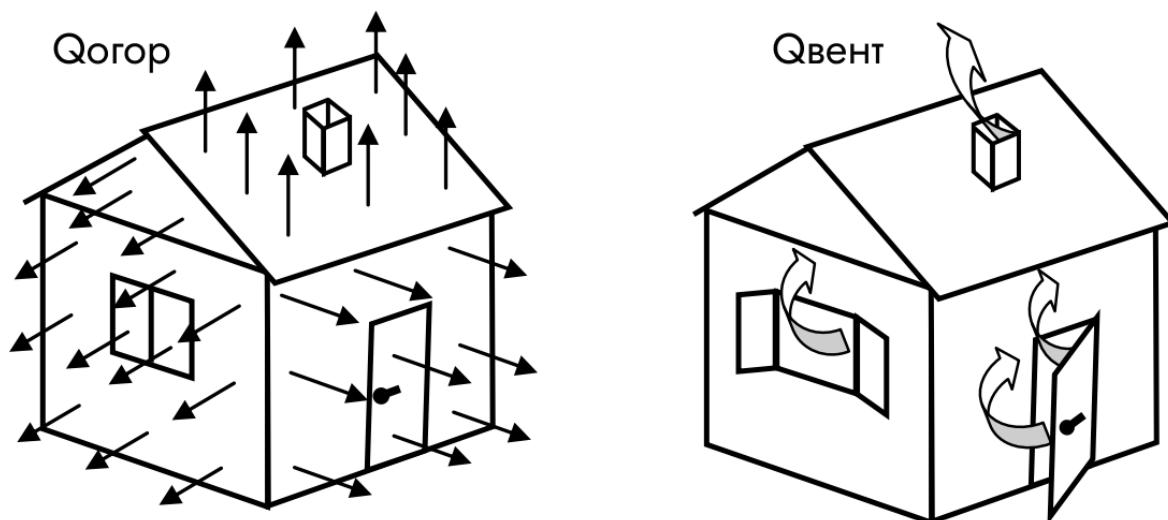


Рис. 1.1. Теплові втрати в навколишнє середовище крізь огорожувальні конструкції та за рахунок вентиляції

Для розрахунку теплового навантаження будівлі застосовується модель, яка враховує теплофізичні властивості матеріалів, кліматичні умови та параметри внутрішнього мікроклімату. Теплове навантаження визначається як сума тепла, що необхідне для компенсації втрат через огорожувальні конструкції, а також тепла, яке потрібне для підтримання встановлених умов у приміщеннях. Розрахунок теплового навантаження зазвичай проводиться для найхолоднішого періоду року, коли різниця між внутрішньою і зовнішньою температурами є максимальною [4].

Процес розрахунку теплового навантаження можна автоматизувати за допомогою інформаційних технологій. Використання спеціалізованих програмних рішень дозволяє спростити обчислення теплових втрат для кожної будівельної конструкції. Розробка таких рішень передбачає інтеграцію алгоритмів, які автоматично враховують параметри матеріалів, кліматичні умови та інші фактори.

Автоматизовані системи здатні враховувати велику кількість змінних, проводити точні обчислення і генерувати звіти, що відповідають вимогам інженерів і проектувальників. Вони дозволяють адаптуватися до змін у вихідних даних, таких як температура зовнішнього середовища, зміни у складі матеріалів чи проектних вимогах.

1.1.1. Втрати теплоти крізь огорожувальні конструкції

Втрата тепла через огорожувальні конструкції розраховується за формулою 1.1.

$$Q_{\text{огор}} = \sum_i \frac{S_i}{R_{0i}} \cdot (t_{\text{вн}} - t_{\text{зовн}}) \cdot n_i \quad (1.1)$$

де, S — площа конструкції, м^2 ;

R_0 — опір теплопередачі огороження, $(\text{м}^2 \cdot \text{C}^0)/\text{Вт}$ [21];

$t_{\text{вн}}$ — внутрішня температура, C^0 ;

$t_{\text{зовн}}$ — зовнішня температура, C^0 ;

n_i — Коефіцієнт для кожної конструкції, який враховує зменшення розрахункової різниці температур, залежить від розташування зовнішньої поверхні огорожувальної конструкції відносно зовнішнього повітря, відповідно до даних, наведених у таблиці 1.1.

Таблиця 1.1

Значення коефіцієнта n

Огорожувальні конструкції	n
Зовнішні стіни, покриття (включаючи вентилявані, які контактують із зовнішнім повітрям), перекриття горищ з дахом із штучних матеріалів, а також над проїздами; перекриття над холодними підпіллями без захисних стінок в північній будівельно-кліматичній зоні.	1
Перекриття над холодними підвалами, що мають вихід до зовнішнього повітря, перекриття горищ із дахом з рулонних матеріалів; перекриття над холодними підпіллями з захисними стінками і холодними поверхами в північній будівельнокліматичній зоні.	0.9
Перекриття над неопалювальними підвалами, що мають світлові отвори	0.75
Перекриття над неопалювальними підвалами без світлових отворів, що розташовані вище рівня землі	0.6
Перекриття над неопалювальними технічними підпіллями, розташованими нижче рівня землі	0.4

Формула 1.1 застосовується для будь-якого огороження, включаючи стіни, покрівлю або підлогу. Основною вимогою є наявність даних про площу термічний опір і різницю температур між середовищами [20].

Для стін і покрівлі різниця температур визначається між внутрішньою температурою та зовнішньою температурою, що є типовою умовою для більшості будівель. У разі, якщо огороження має додатковий шар утеплення, термічний опір визначається як сума термічних опор усіх шарів конструкції. Для утеплених огорожувальних конструкцій враховується матеріал утеплювача та його товщина [15]. Тобто для таких випадків зовнішню температуру варто вважати як температуру підвалу або ґрунту.

Опір теплопередачі R_0 розраховується для кожної огорожувальної поверхні за формулою 1.2.

$$R_0 = \frac{1}{\alpha_B} + R_K + \frac{1}{\alpha_3} \quad (1.2)$$

де, R_K – термічний опір огорожувальної конструкції, $(\text{м}^2 \cdot \text{C}^0)/\text{Вт}$;

α_B і α_3 – коефіцієнти тепловіддачі на внутрішній і зовнішній поверхнях огорожувальних конструкцій, $\text{Вт}/(\text{м}^2 \cdot \text{C}^0)$, (табл. 1.2) [8].

Таблиця 1.2

Значення коефіцієнтів тепловіддачі зовнішньої α_3 та внутрішньої α_B поверхонь огорожувальних конструкцій

Тип конструкції	Коефіцієнт тепловіддачі, $\text{Вт}/(\text{м}^2 \cdot \text{C}^0)$	
	α_3	α_B
Зовнішні стіни, дахи, покриття, перекриття над проїздами, плоскі та з ребрами, при відношенні висоти ребра h до відстані між гранями b сусідніх ребер $h/b \leq 0,3$ $h/b > 0,3$	8,7 7,6	23 23
Перекриття горищ та холодних підвалів	8,7	12
Перекриття над холодними підвалами та технічними поверхами, що розташовані нижче рівня землі	8,7	6
Вікна, балконні двері, вітражі та світлопрозорі фасадні системи	8,0	23

Термічний опір огорожувальних конструкцій розраховується для кожного шару матеріалу [5, с. 11 – 20] (Рис. 1.2) і визначається за формулою 1.3.

$$R = \frac{\delta}{\lambda} \quad (1.3)$$

де, δ — товщина матеріалу,

λ — теплопровідність матеріалу.

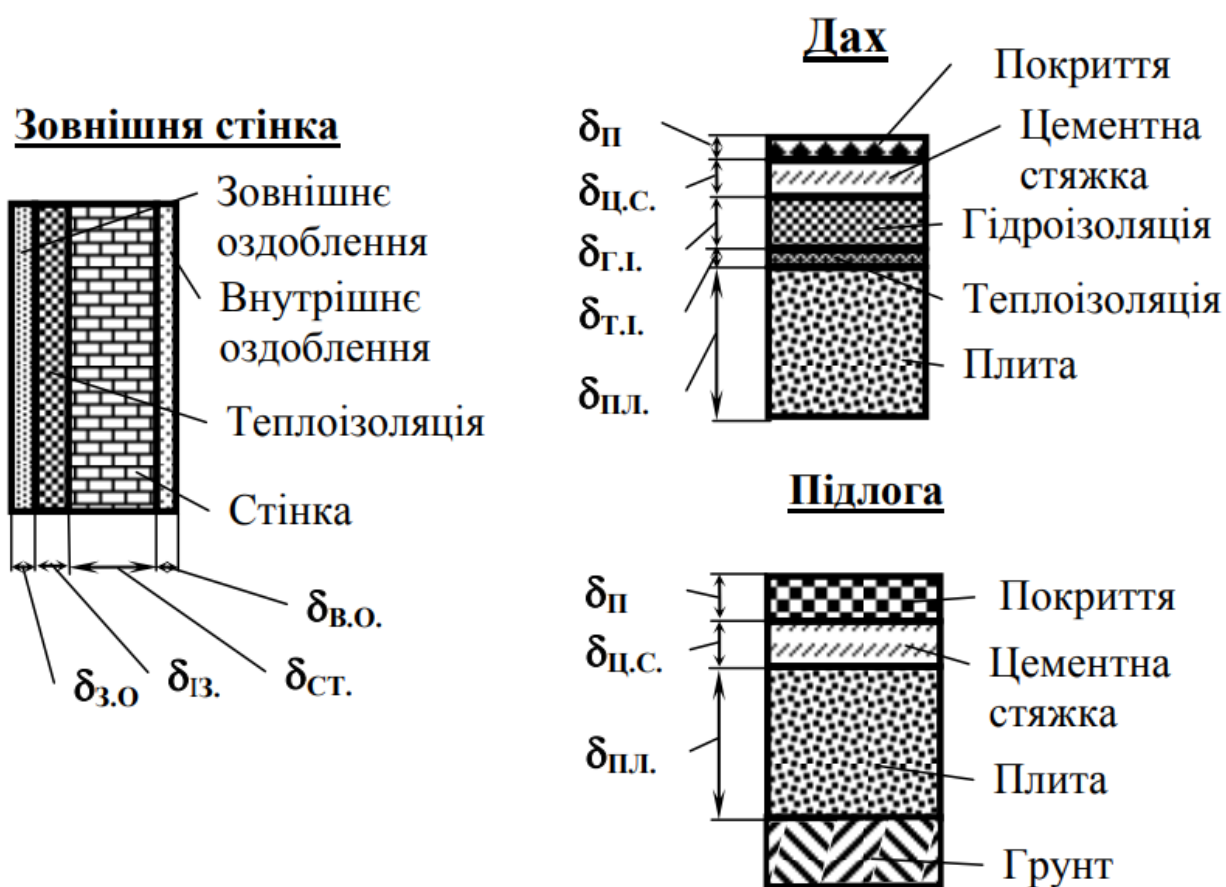


Рис. 1.2. Вертикальні розрізи зовнішньої стінки, даху та підлоги

1.1.2. Теплове навантаження на вентиляцію

Вентиляція являє собою сукупність заходів та технічних засобів, що застосовуються для організації повітрообміну з метою забезпечення визначених параметрів повітряного середовища в приміщеннях та на робочих місцях (табл. 1.3). Функціонування систем вентиляції спрямоване на підтримання допустимих метеорологічних характеристик у приміщеннях різного функціонального призначення [19].

Таблиця 1.3

Мінімальна витрата, м³ /год, зовнішнього повітря на 1 людину, яка перебуває в помешканні понад 2 години

Будівлі	Помешкання	
	З природною вентиляцією	Без природної вентиляції
Виробничі	30	60
Громадського та адміністративного призначення	40	60
Житлові з загальною площею на 1 людину понад 20 м ²	30	60

Можна приймати за типовими або індивідуальними проектами, які найбільше відповідають характеристикам конкретних будівель. У разі відсутності даних про максимальні витрати теплоти на вентиляцію, потребу в тепловій енергії, Вт, розраховують за формулою 1.4.

$$Q_{\text{вент}} = K \cdot Q_{\text{огор}} \quad (1.4)$$

де, К (коефіцієнт) = 0,4 для будівель зведених до 1985 року, та К = 0,6 – побудованих після 1985 року.

Вентиляційні тепловтрати розраховуються за формулою 1.5, коли відомо призначення будівлі та його об'єм.

$$Q_{\text{вент}} = c \cdot \rho \cdot L \cdot (t_{\text{вн}} - t_{\text{зовн}}) \quad (1.5)$$

де, с — теплоємність повітря,

ρ — густина повітря,

L — об'єм повітря, що оновлюється за одиницю часу [5, с. 37 – 40].

Об'єм повітря L, що оновлюється за одиницю часу, визначається з урахуванням норм повітрообміну для приміщення. Щоб його визначити можна використати кратність повітрообміну. Кратність повітрообміну показує, скільки разів за годину повітря в приміщенні має бути повністю оновлене. Цей параметр залежить від типу приміщення (житлове, офісне, промислове тощо) Він береться із санітарних норм або будівельних стандартів або розраховується за формулою (формула 1.6).

$$L = n \cdot V \quad (1.6)$$

де, n — кратність повітрообміну (раз/год, згідно з нормами),

V — об'єм приміщення (м^3).

Тобто формули 1.5 та 1.6 можна звести до однієї (формула 1.7).

$$Q_{\text{вент}} = n \cdot c \cdot V \cdot \rho \cdot (t_{\text{вн}} - t_{\text{зовн}}) \quad (1.7)$$

У випадку, коли будівля містить приміщення різного призначення, які мають відмінності в питомих вентиляційних характеристиках або розрахункових температурах внутрішнього повітря, тепловий потік на вентиляцію розраховують для кожної частини окремо, після чого отримані значення підсумовують.

1.1.3. Автоматизація розрахунків за допомогою інформаційних технологій

Для автоматизації процесу розрахунків теплового навантаження на будівлі використовуються інформаційні технології, які базуються на застосуванні алгоритмів, математичних моделей та числових методів. Ці технології дозволяють зменшити трудомісткість і час, необхідний для виконання розрахунків, а також знизити ймовірність помилок, пов'язаних з ручними обчисленнями. Основними етапами автоматизації є формалізація процесу розрахунку, розробка програмних засобів та інтеграція алгоритмів у програмне забезпечення [17].

Автоматизація розрахунків теплового навантаження включає в себе використання математичних моделей, описаних вище, для обчислення теплових втрат через огорожувальні конструкції, теплового навантаження на вентиляцію, а також інших елементів будівлі, які впливають на енергоефективність. Для цього застосовуються як аналітичні, так і числові методи моделювання, що дозволяє враховувати численні фактори, такі як зовнішні температурні коливання, характер вентиляційних систем та теплові характеристики матеріалів.

Алгоритми автоматизованого, базуються на виконанні окремих обчислень для кожного елемента будівлі: стін, покрівель, перекриттів та вентиляційних систем. Для кожної з цих частин будівлі обчислюються теплові втрати або теплове навантаження за допомогою відповідних математичних формул, що описують теплопередачу через матеріали, а також теплообмін з навколишнім середовищем. Розрахунки проводяться поетапно для кожного елемента, після чого результати сумуються, щоб визначити загальне теплове навантаження для будівлі. Такий підхід дозволяє враховувати специфічні характеристики кожного конструктивного елемента та його взаємодію з іншими частинами будівлі.

Для реалізації алгоритмів автоматизації розрахунків теплового навантаження активно застосовуються сучасні програмні засоби. Одним з найбільш поширених підходів є використання високорівневих мов програмування, таких як Python, C++ чи Java [18], у поєднанні з числовими бібліотеками для обробки даних та виконання математичних операцій. В Python, наприклад, можуть використовуватися бібліотеки NumPy, SciPy, а також бібліотеки для побудови математичних моделей і вирішення рівнянь, такі як SymPy.

Крім того, для представлення результатів розрахунків у вигляді звітів використовуються засоби для автоматичного формування текстових звітів у форматах PDF або Word. У звіт вноситься опис температурних профілів та теплових навантажень для кожної будівельної конструкції, де вказуються розраховані значення теплових втрат або навантажень, відповідно до кожного елемента (стіни, покрівля, перекриття, вентиляція).

1.2. Методологічні підходи до дослідження теплового навантаження

Максимальне теплове навантаження є сукупністю втрат тепла через огорожувальні конструкції і вентиляцію. Воно визначає пікову кількість теплової енергії, яка потрібна для компенсації всіх тепловтрат будівлі за одиницю часу. Це значення зазвичай розраховується для найнижчої можливої

температури зовнішнього середовища в даній кліматичній зоні. Для аналізу теплового навантаження також враховують особливості експлуатації будівлі.

Методологічні підходи до дослідження теплового навантаження базуються на застосуванні математичних моделей, що враховують фізичні характеристики будівлі, кліматичні умови, температурні параметри та експлуатаційні особливості приміщень.

1.2.1. Розрахунок теплового навантаження будівлі

Розрахунок теплового навантаження починається зі збору вихідних даних про будівлю. До таких даних належать площа приміщень, площа зовнішніх стін, висота стін, матеріал стін, тип приміщення за функціональним призначенням, а також температури внутрішнього середовища та зовнішнього повітря. Ці параметри є основою для визначення термічного опору огорожувальних конструкцій і обчислення вентиляційних втрат, що здійснюється на основі нормативних формул, розглянутих у попередньому пункті.

Методологія передбачає розрахунок втрат тепла через огорожувальні конструкції, який виконується з урахуванням теплофізичних властивостей матеріалів, геометричних характеристик будівлі та різниці температур між внутрішнім і зовнішнім середовищем [6]. Термічний опір огорожувальних конструкцій обчислюється на основі довідкових даних, а тепловтрати оцінюються за допомогою інтеграції цих значень у математичну модель. Зокрема, для кожного типу конструкцій застосовуються відповідні коефіцієнти теплопровідності, що відображають теплофізичні властивості матеріалів. Розрахунок враховує також геометричні параметри, такі як товщина стін, площа огорожувальних елементів [22], що дозволяє точно обчислити потік тепла через конструкції в умовах зовнішніх температур.

Розрахунок вентиляційних втрат базується на визначенні кількості повітря, що обмінюється в приміщенні, з використанням нормативної кратності повітрообміну, залежної від типу приміщення [2]. Параметри температури

повітря, його теплоємності та густини враховуються при обчисленні енергетичних втрат, пов'язаних із вентиляцією. Припускається, що повітрообмін відповідає нормам, а температурні показники залишаються стабільними, що спрощує розрахунок і підвищує практичність методології. Оскільки вентиляційні втрати є значним компонентом теплового навантаження, їх правильний облік забезпечує точність загальної оцінки теплових втрат, що необхідно для коректного визначення енергетичних потреб будівлі [1].

Тепло, яке виділяється людьми та технічним обладнанням (освітлення, комп'ютери, побутова техніка), також є частиною теплового навантаження будівлі. Для людей зазвичай використовуються стандартні величини теплових потоків, що залежать від кількості осіб та їхньої активності.

Основні етапи розрахунку включають:

- Визначення площі огорожувальних конструкцій (стіни, покрівля, підлога).
- Розрахунок термічного опору матеріалів.
- Обчислення вентиляційних втрат за допомогою визначених нормативів та коефіцієнтів.
- Визначення теплового навантаження через інтеграцію втрат через конструкції та вентиляцію в одну загальну модель.
- Оцінка теплових потоків від людей в приміщенні

Реалізація методології передбачає створення програмного забезпечення, яке забезпечує автоматизацію розрахунків теплового навантаження. Уведення вихідних даних користувачем здійснюється через інтуїтивно зрозумілий інтерфейс, який дозволяє вказати площу приміщення, площу зовнішніх стін, висоту стін, матеріал стін, тип приміщення та температури внутрішнього й зовнішнього середовища. Система автоматично виконує обчислення теплового навантаження відповідно до описаних моделей і надає результати у вигляді підсумкових значень для втрат тепла через огорожувальні конструкції та вентиляцію.

Розрахунок теплового навантаження будівель, описаний у методології, доцільно реалізувати у вигляді спеціалізованого програмного забезпечення. Таке програмне забезпечення має виконувати всі етапи обчислень автоматично, ґрунтуючись на вихідних даних, що вводяться користувачем.

1.2.2. Використання комп'ютерних технологій для аналізу теплового навантаження

У сучасних умовах для аналізу теплового навантаження будівель активно використовуються комп'ютерні технології. Основним інструментом для реалізації таких розрахунків є спеціалізоване програмне забезпечення, яке автоматизує процеси збору, обробки і аналізу вихідних даних.

Одним із варіантів реалізації є використання мови програмування Python. Python є кросплатформеним рішенням [14], яке дозволяє створювати програми, що працюють на різних операційних системах. Мова підтримує широкий спектр бібліотек для математичних розрахунків, роботи з текстом та створення графічних інтерфейсів.

Для виконання необхідних обчислень можуть використовуватися наступні бібліотеки Python:

- NumPy
- Pandas
- Matplotlib або Seaborn
- tkinter або PyQt

NumPy є бібліотекою для числових розрахунків, що забезпечує виконання математичних операцій над масивами та матрицями. Вона широко використовується при розрахунках теплових втрат через огорожувальні конструкції та вентиляційні втрати. Завдяки своїм можливостям працювати з великими обсягами даних, NumPy дозволяє швидко виконувати обчислення, що оптимізує процес моделювання [29].

Для обробки структурованих даних застосовується бібліотека Pandas. Вона дозволяє ефективно працювати з таблицями та великими наборами даних,

такими як площі будівель, матеріали та температури. Pandas надає інструменти для маніпулювання, фільтрації та агрегації даних, що необхідно для підготовки інформації до подальших розрахунків теплового навантаження.

Візуалізація результатів розрахунків здійснюється за допомогою бібліотек Matplotlib і Seaborn. Matplotlib дозволяє створювати графіки, діаграми та гістограми, що забезпечують наочне представлення результатів [11]. Seaborn доповнює Matplotlib [25], надаючи додаткові функції для створення більш складних візуалізацій, зокрема для аналізу взаємозв'язків між параметрами [28].

Для розробки графічних інтерфейсів користувача використовуються бібліотеки Tkinter та PyQt. Tkinter є стандартною бібліотекою Python [34], що дозволяє швидко створювати прості інтерфейси. PyQt забезпечує більшу гнучкість і можливості для розробки складних інтерфейсів, які включають інтерактивні елементи для введення даних, що використовуються під час розрахунків [30].

Застосунок повинен передбачати введення вихідних даних через просту форму, що включає текстові поля та випадаючі списки для вибору параметрів, таких як тип матеріалів або тип конструкції. На основі введених даних програма виконує обчислення теплових втрат через огорожувальні конструкції, вентиляцію та додає їх у звіт.

Результатом роботи програмного забезпечення є звіт у текстовому форматі, який може бути збережений у файл або відображений на екрані. У звіті зазначаються введені параметри, результати розрахунків окремо для кожного компонента теплових втрат, а також загальне теплове навантаження будівлі.

РОЗДІЛ 2. ПРАКТИЧНА РЕАЛІЗАЦІЯ РОЗРОБКИ ЗАСТОСУНКУ ДЛЯ СТВОРЕННЯ ПРОФІЛЮ ТЕПЛОВОГО НАВАНТАЖЕННЯ

2.1. Аналіз вхідних даних для побудови профілю теплового навантаження

Аналіз підходів до розрахунку теплового навантаження будівлі свідчить про необхідність визначення характеристик огорожувальних конструкцій. До таких характеристик належать тип конструкції, її площа, товщина шарів та матеріали, з яких ці шари виготовлені. Використовуючи зазначені параметри разом із табличними коефіцієнтами для відповідних матеріалів і типів конструкцій, можна провести точний розрахунок теплової потужності та сформувати профіль теплового навантаження будівлі.

Для візуалізації взаємозв'язку між вхідними та вихідними даними застосунку створено IDEF0-діаграму, яка відображає основні етапи процесу [10]. Ця діаграма представлена на Рис. 2.1.

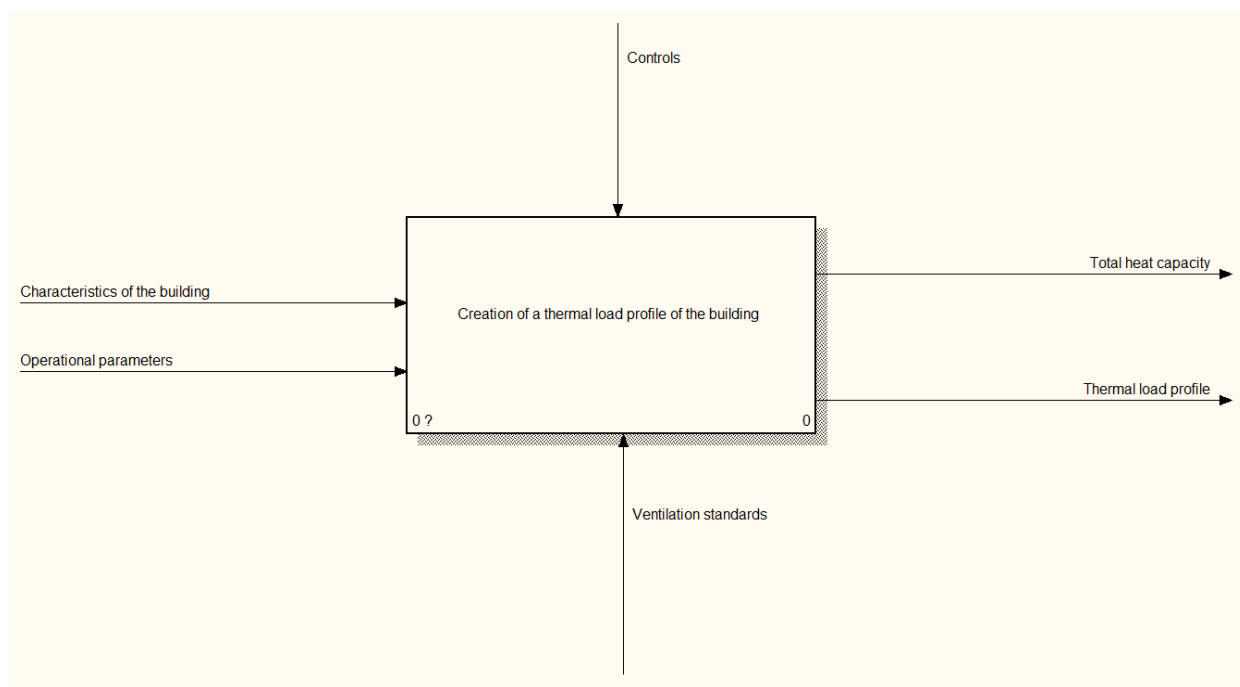


Рис. 2.1. IDEF0-діаграма процесу формування профілю теплового навантаження будівлі

На рисунку продемонстровано модель процесу створення профілю теплового навантаження будівлі. У центральному блоці зазначено основний об'єкт процесу – Creation of a thermal load profile of the building. Як вхідні дані використовуються характеристики будівлі, параметри її експлуатації, норми вентиляції та регульовальні показники. Результатом процесу є загальна теплова потужність будівлі (Total heat capacity) та її профіль теплового навантаження (Thermal load profile), що використовуються для аналізу енергетичної ефективності будівлі та проектування інженерних систем.

2.2. Розробка алгоритму побудови профілю теплового навантаження

Для визначення загального алгоритму побудови профілю теплового навантаження будівлі була проведена декомпозиція процесу у вигляді DFD-діаграми [7]. На Рис. 2.2 наведено відповідну діаграму, яка відображає логічну послідовність етапів виконання операцій для формування теплового навантаження будівлі.

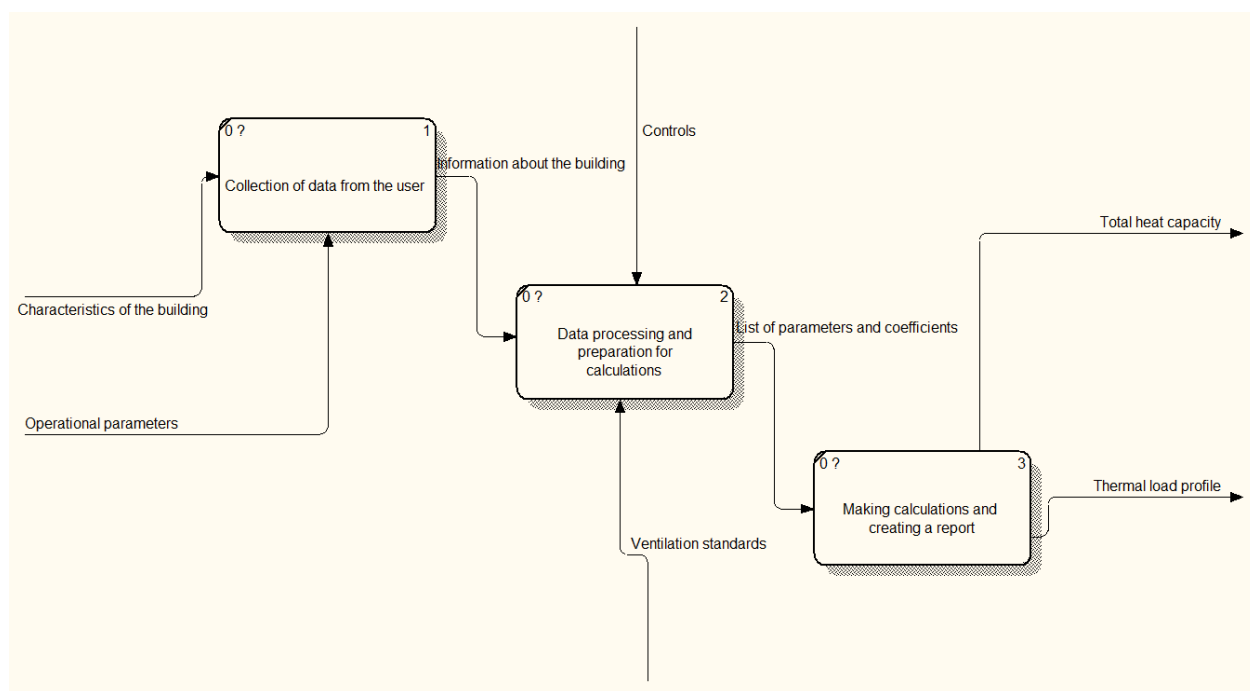


Рис. 2.2. DFD-діаграма процесу створення профілю теплового навантаження будівлі

Модель процесу складається з трьох основних функціональних блоків.

Перший етап, *Collection of data from the user*, забезпечує збір необхідних даних про будівлю. Вхідними параметрами цього етапу є характеристики будівлі, що включають дані про типи огорожувальних конструкцій, їхні площі, матеріали і товщини, а також параметри експлуатації, такі як внутрішня і зовнішня температура. На виході формується структурована інформація про будівлю (*Information about the building*), яка передається на наступний етап.

Другий етап, *Data processing and preparation for calculations*, виконує обробку отриманих даних. Вхідними параметрами цього блоку є структурована інформація про будівлю, регульовальні параметри (*Controls*) та нормативні показники вентиляції (*Ventilation standards*). На основі цих даних розраховуються і формуються параметри й коефіцієнти, необхідні для виконання подальших розрахунків. Результатом є сформований список параметрів та коефіцієнтів (*List of parameters and coefficients*).

Третій етап, *Making calculations and creating a report*, відповідає за виконання теплотехнічних розрахунків та формування звіту. Вхідними даними є список параметрів і коефіцієнтів. На цьому етапі розраховується загальна теплова потужність будівлі (*Total heat capacity*), а також створюється профіль теплового навантаження (*Thermal load profile*). Ці результати слугують основою для оцінки теплових характеристик будівлі.

Розглянемо алгоритми дещо детальніше. Взаємодія з користувачем відбувається через графічний інтерфейс користувача (UI), що включає кнопки та діалогові вікна з формами для введення необхідних даних. Користувач може створювати та редагувати конструкції огорожень і рівнів, заповнюючи відповідні поля в діалогових вікнах. Алгоритм отримання даних від користувача для створення огорожень та рівнів, виконаний в draw.io [24], наведено на блок-схемі (Рис. 2.3).

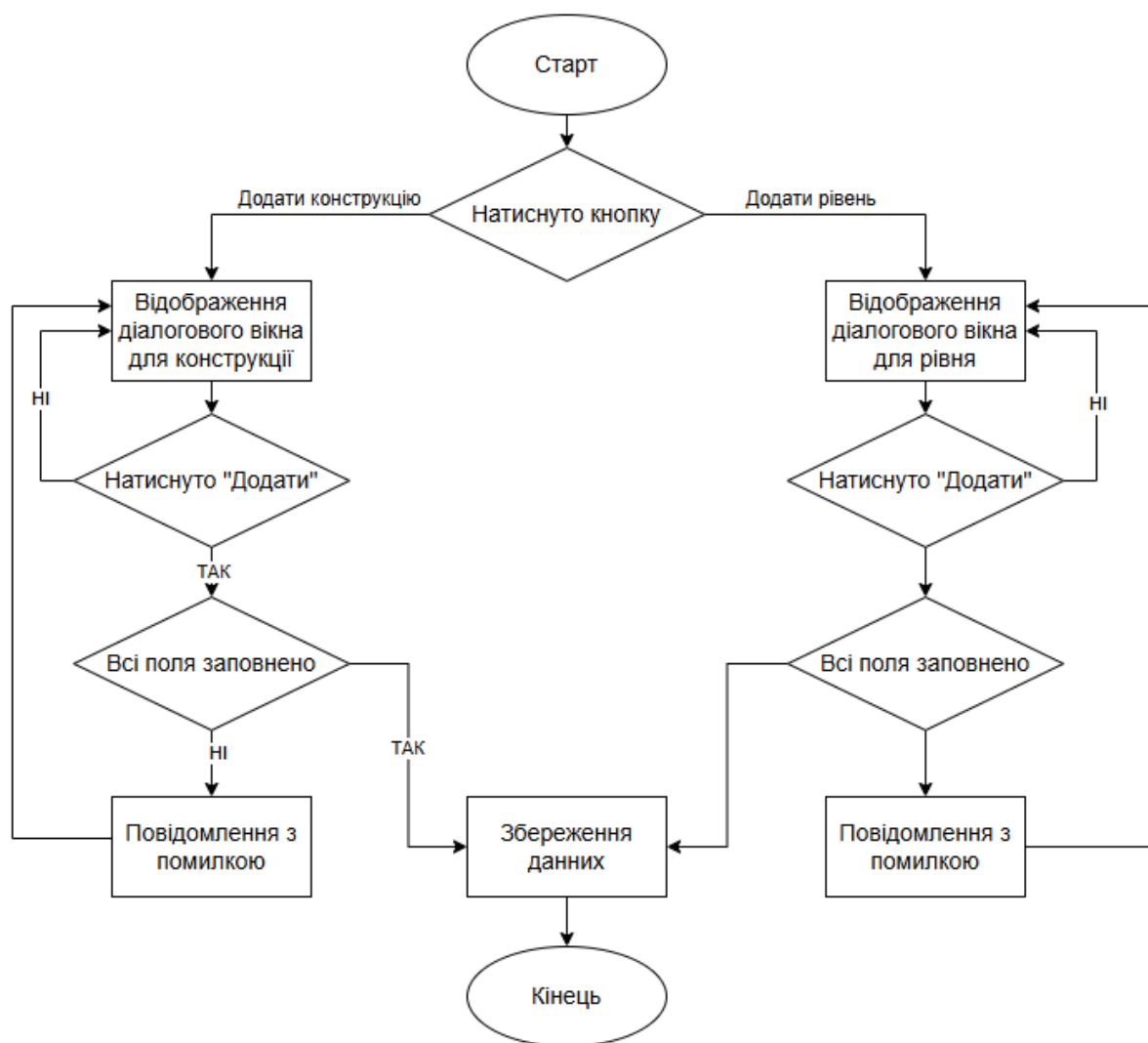


Рис. 2.3. Алгоритм отримання даних від користувача

На схемі показано основні етапи взаємодії користувача з системою. Спочатку користувач натискає кнопку для створення конструкції або рівня. Потім відкривається діалогове вікно, в якому необхідно заповнити всі поля. У разі, якщо всі поля заповнені коректно, дані зберігаються, і діалогове вікно закривається. Якщо введені дані не відповідають вимогам (наприклад, поля не заповнені або введені некоректні значення), користувач отримує відповідне повідомлення про помилку. В результаті взаємодії з користувачем система збирає необхідні дані для подальших розрахунків і побудови профілю теплового навантаження.

Далі розглянемо процес підготовки даних. Цей етап завершує перевірку введених даних, що розпочалася під час введення, та формує структуру даних на основі класів, створених для зберігання інформації про будівлю. Після цього

дані структуруються у формат, придатний для подальших розрахунків і збереження. Протягом цього процесу проводиться перевірка на коректність введених значень, і у разі виявлення помилок процес завершується. У разі успішної перевірки створюється список у форматі JSON, що містить всі необхідні параметри для подальших обчислень.

Блок-схема цього процесу наведена на Рис. 2.4.

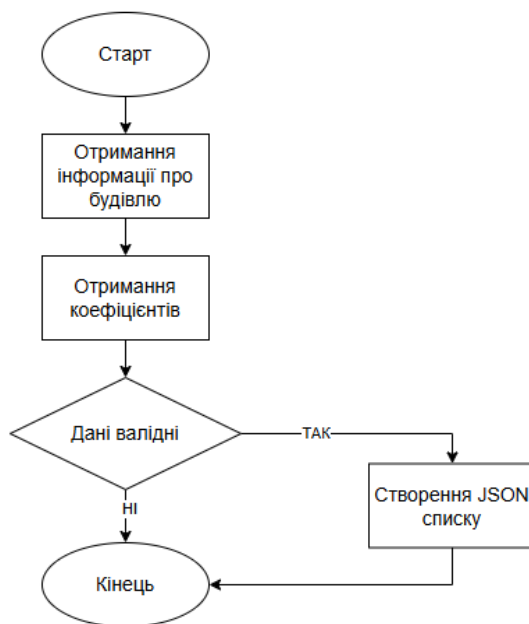


Рис. 2.4. Блок-схема підготовки даних для обчислень

На схемі показано, як відбувається обробка даних після введення: спочатку отримується інформація про будівлю та необхідні коефіцієнти, потім перевіряється валідність даних. Якщо дані некоректні, процес завершується. Якщо ж перевірка пройшла успішно, створюється відповідна структура даних у форматі JSON, готова для подальших операцій.

Розглянемо алгоритм для розрахунку та створення звіту. На цей етап потрапляють уже валідні дані, тобто ті, що пройшли перевірку на коректність. Процес розрахунку включає кілька етапів: обчислення теплового навантаження для кожної конструкції, розрахунок термічного опору кожного рівня конструкції, а також визначення загального теплового навантаження, що враховує всі конструкції будівлі. На основі цих обчислень формується звіт, який містить усі необхідні дані для подальшого аналізу.

Алгоритм, що використовується для виконання цих розрахунків, зображено на Рис. 2.5.

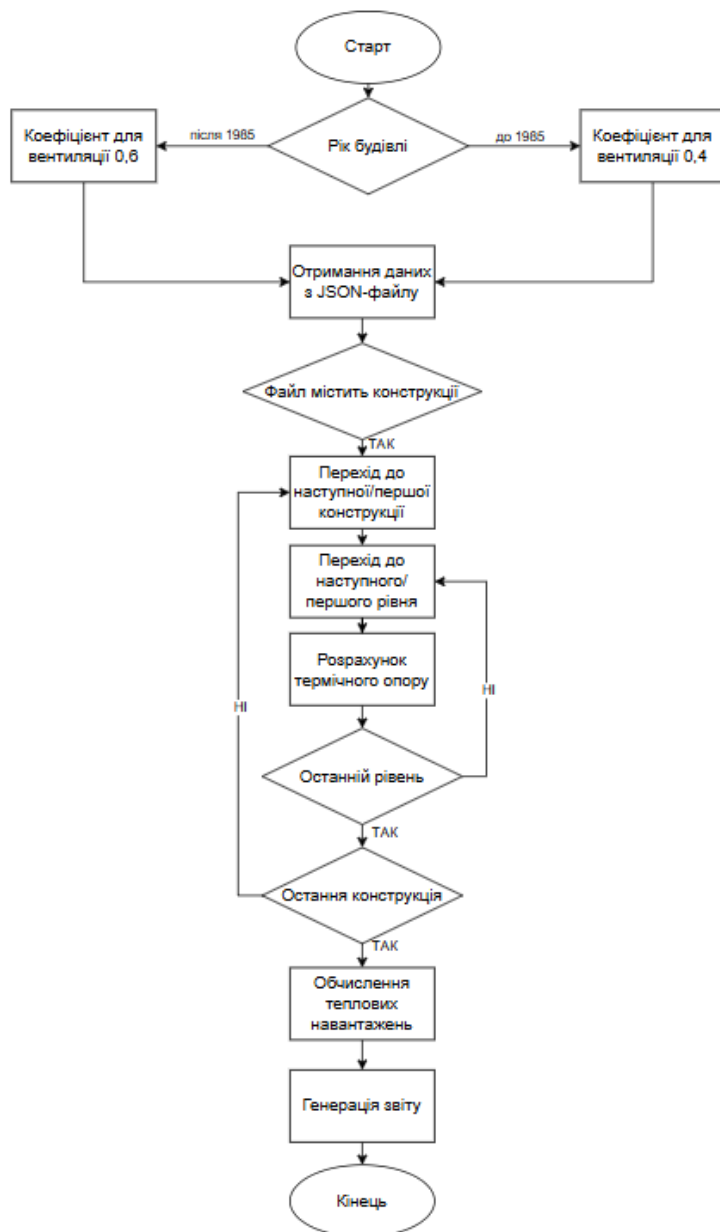


Рис. 2.5. Алгоритм розрахунку теплового навантаження та формування звіту

Алгоритм починається з вибору або введення року побудови будівлі, після чого зчитуються дані з файлу, що містить параметри конструкцій. Для кожної конструкції визначається термічний опір, що залежить від товщини матеріалів і їх теплопровідності. Розрахунок теплового навантаження кожної конструкції проводиться з урахуванням її площі, внутрішньої та зовнішньої температури, а також коефіцієнтів теплопередачі. Далі обчислюється загальне

теплове навантаження для всієї будівлі, враховуючи також навантаження від вентиляції, яке коригується в залежності від року будівлі.

Завершенням алгоритму є формування звіту у вигляді PDF-документу, який містить усі необхідні дані про результати розрахунків, а також деталі про кожну конструкцію.

2.3. Реалізація програмного забезпечення

Програмне забезпечення, яке розробляється в рамках цього проекту, призначене для обчислення теплового навантаження будівлі на основі даних про конструкції. Воно містить кілька вкладок, що виконують різні функції. Користувач має змогу ввести необхідні дані для розрахунку, та згенерувати звіт на основі введених даних.

2.3.1. Створення класів додатку

Для реалізації об'єктно-орієнтованого підходу [23] в програмі було створено класи, які відповідають за моделювання основних компонентів будівлі: огорожувальних конструкцій, рівнів конструкцій та самої будівлі. На Рис. 2.6 наведено код для створення класу Building, який представляє будівлю в цілому.

```
class Building:
    def __init__(self):
        self.fences = []
        self.ventilations = []

    def add_fence(self, fence):
        self.fences.append(fence)

    def add_ventilation(self, ventilation):
        self.ventilations.append(ventilation)

    def calculate_total_heat_loss(self):
        total_fence_resistance = sum(fence.calculate_total_thermal_resistance() for fence in self.fences)
        total_ventilation_loss = sum(ventilation.calculate_heat_loss() for ventilation in self.ventilations)
        return total_fence_resistance + total_ventilation_loss
```

Рис. 2.6. Код класу Building

Клас `Building` призначений для моделювання будівлі в цілому. Він містить два основних атрибути: `fences` та `ventilations`, які зберігають відповідно список огорожувальних конструкцій та систем вентиляції. Ці атрибути є порожніми списками на момент створення об'єкта. Клас також містить методи для додавання огорожувальних конструкцій (`add_fence`) та вентиляційних систем (`add_ventilation`). Ці методи дозволяють доповнювати будівлю новими елементами, що дає змогу моделювати складні будівельні конструкції з кількома компонентами.

Основною функцією класу є метод `calculate_total_heat_loss`, який обчислює загальне теплове навантаження будівлі. Для цього здійснюється підсумовування теплових втрат від усіх огорожувальних конструкцій і систем вентиляції. Кожна огорожувальна конструкція має метод `calculate_total_thermal_resistance`, що відповідає за розрахунок теплового опору конструкції, а кожна система вентиляції має метод `calculate_heat_loss`, який визначає теплові втрати від вентиляції. Результати цих обчислень підсумовуються для отримання загального теплового навантаження на будівлю.

Далі реалізуємо клас для огорожувальної конструкції, це клас `Fence`, код наведено на Рис. 2.7.

```
class Fence:
    def __init__(self, fence_type, exterior_temperature, interior_temperature, area, coef=1.0, alpha_interior=0.0, alpha_exterior=0.0):
        self.fence_type = fence_type
        self.coef = coef
        self.area = area
        self.interior_temperature = interior_temperature
        self.exterior_temperature = exterior_temperature
        self.alpha_interior = alpha_interior
        self.alpha_exterior = alpha_exterior
        self.levels = []

    def add_level(self, level):
        self.levels.append(level)

    def calculate_total_thermal_resistance(self):
        total_resistance = sum(level.get_thermal_resistance() for level in self.levels)
        return total_resistance
```

Рис. 2.7. Код класу `Fence`

Клас `Fence` призначений для моделювання огорожувальних конструкцій, що складаються з кількох рівнів. Клас містить кілька атрибутів, які характеризують огорожувальну конструкцію:

— `fence_type` — тип огорожувальної конструкції (наприклад, стіна, вікно, дах тощо).

- coef – коефіцієнт, що визначає особливості теплової поведінки конструкції (за замовчуванням дорівнює 1.0).
- area – площа огорожувальної конструкції.
- interior_temperature – температура всередині будівлі.
- exterior_temperature – температура зовнішнього середовища.
- alpha_interior та alpha_exterior – коефіцієнти теплопередачі для внутрішньої та зовнішньої поверхні відповідно.

Для додавання рівнів конструкції передбачено метод `add_level`, який дозволяє додавати до конструкції нові рівні матеріалу, кожен з яких може мати власну характеристику теплового опору [3]. Цей метод допомагає змодельовати багатошарові конструкції, такі як стіни з декількох матеріалів.

Далі реалізуємо клас для рівня конструкції, це клас `Level`, код наведено на Рис. 2.8.

```
class Level:
    def __init__(self, material, thickness):
        self.material = material
        self.thickness = thickness

    def get_thermal_resistance(self):
        return self.thickness / self.material.thermal_conductivity
```

Рис. 2.8. Код класу `Level`

Клас `Level` призначений для моделювання окремих рівнів матеріалу в огорожувальних конструкціях. Кожен рівень характеризується матеріалом, з якого він виготовлений, та його товщиною. Клас містить два основних атрибути:

- material – матеріал рівня, що містить властивості теплопровідності (наприклад, бетон, дерево, цегла).
- thickness – товщина матеріалу рівня.

Метод `get_thermal_resistance` визначає тепловий опір цього рівня, використовуючи його товщину та теплопровідність матеріалу. Тепловий опір обчислюється за формулою 1.3.

2.3.2. Реалізація допоміжних методів

Для виконання проєкту створені кілька допоміжних методів, які забезпечують завантаження та збереження даних з файлів у форматі JSON [9]. Ці методи використовуються для обробки інформації, необхідної для розрахунків теплового навантаження будівлі. Далі наводиться опис кожного з методів.

```
def load_json_data(filename):
    try:
        with open(os.path.join('json_data', filename), 'r', encoding='utf-8') as file:
            return json.load(file)
    except FileNotFoundError:
        print(f"Файл {filename} не знайдений.")
        return []
    except json.JSONDecodeError:
        print(f"Помилка при зчитуванні даних з файлу {filename}.")
        return []
```

Рис. 2.9. Код методу load_json_data

Метод load_json_data відповідає за завантаження даних з файлу у форматі JSON. Він приймає один параметр – ім'я файлу. Після цього метод намагається відкрити файл, зчитати його вміст та перетворити в Python об'єкт за допомогою json.load() [27]. Якщо файл не знайдено або виникає помилка при декодуванні JSON, метод виводить відповідне повідомлення про помилку та повертає порожній список. Така обробка помилок дозволяє забезпечити коректну роботу програми навіть у разі проблем з файлом.

Метод get_enclosure_types завантажує дані з файлу enclosure_types.json, використовуючи попередньо описаний метод load_json_data. Після завантаження даних метод формує список рядків, який містить типи огорожувальних конструкцій разом з їх коефіцієнтами та значеннями теплових характеристик. Цей метод повертає список форматуваних рядків, який можна використовувати для подальшої обробки або відображення в інтерфейсі користувача.

```
def save_data_fence(building, filename):
    data = {
        "fences": [
            {
                "fence_type": fence.fence_type,
                "coef": fence.coef,
                "area": fence.area,
                "interior_temperature": fence.interior_temperature,
                "exterior_temperature": fence.exterior_temperature,
                "alpha_interior": fence.alpha_interior,
                "alpha_exterior": fence.alpha_exterior,
                "levels": [
                    {
                        "material": level.material.material_type,
                        "thermal_conductivity": level.material.thermal_conductivity,
                        "thickness": level.thickness
                    }
                    for level in fence.levels
                ]
            }
            for fence in building.fences
        ]
    }

    try:
        with open(filename, 'w') as json_file:
            json.dump(data, json_file, indent=4)
        print(f"Дані збережено в файл '{filename}'.")
    except Exception as e:
        print(f"Помилка при збереженні даних: {e}")
```

Рис. 2.10. Код методу save_data_fence

Метод save_data_fence призначений для збереження даних про огорожувальні конструкції будівлі в JSON файл. Він приймає два параметри: об'єкт будівлі та ім'я файлу. Спочатку метод збирає необхідні дані про кожну конструкцію, зберігаючи їх у відповідному форматі. Для кожної конструкції зберігаються тип, коефіцієнти, температура та інші характеристики, а також деталі кожного рівня матеріалу. Потім ці дані зберігаються у JSON файл. Якщо виникає помилка під час збереження, виводиться повідомлення з описом помилки.

```
def load_data_fence(building, filename):
    try:
        with open(filename, 'r') as json_file:
            data = json.load(json_file)
            for fence_data in data.get('fences', []):
                fence = Fence(
                    fence_data['fence_type'],
                    fence_data['exterior_temperature'],
                    fence_data['interior_temperature'],
                    fence_data['area'],
                    fence_data['coef'],
                    fence_data['alpha_interior'],
                    fence_data['alpha_exterior']
                )
                for level_data in fence_data['levels']:
                    material = Material(level_data['material'], level_data['thermal_conductivity'])
                    level = Level(material, level_data['thickness'])
                    fence.add_level(level)
                building.add_fence(fence)
            print(f"Дані завантажено з файлу '{filename}'.")
    except Exception as e:
        print(f"Помилка при завантаженні даних: {e}")
```

Рис. 2.11. Код методу load_data_fence

Метод `load_data_fence` відповідає за завантаження даних з файлу JSON та відновлення об'єктів конструкцій у програмі. Він приймає два параметри: об'єкт будівлі та ім'я файлу. Після зчитування даних з файлу метод створює відповідні об'єкти конструкцій, рівнів матеріалів та додає їх до будівлі. Якщо виникає помилка при завантаженні даних, метод виводить повідомлення про помилку.

2.3.3. Реалізація центрального файлу програми

Центральний файл програми (`app.py`) відповідає за ініціалізацію головного вікна додатку, створення вкладок та запуск програми. Далі описано, як організована реалізація цього файлу. Кожен метод та частина коду пояснюються окремо для кращої зрозумілості.

```
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QTabWidget, QWidget
from ui.main_ui import Ui_MainWindow
from ui.fence_ui import Ui_FenceTab
from ui.summary_ui import Ui_SummaryTab
from ui.info_ui import Ui_InfoTab
```

Рис. 2.11. Імпорт бібліотек

На початку файлу імпортуються необхідні бібліотеки. Основною бібліотекою є `PyQt5`, яка використовується для створення графічного інтерфейсу користувача (GUI) [12]. Також імпортуються класи з файлів UI, які були створені для кожної вкладки додатку.

```
class BuildingApp(QMainWindow):
    def __init__(self):
        super().__init__()

        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        self.setup_tabs()
```

Рис. 2.12. Клас `BuildingApp`

У цьому фрагменті визначено клас `BuildingApp`, який є головним вікном програми. Клас успадковується від `QMainWindow` [38], що дозволяє

створювати основне вікно додатку. В конструкторі класу викликається метод `setupUi()`, що ініціалізує графічний інтерфейс головного вікна, використовуючи об'єкт `Ui_MainWindow`, який відповідає за створення основного інтерфейсу користувача.

Після ініціалізації основного інтерфейсу викликається метод `setup_tabs()`, який відповідає за налаштування вкладок у вікні.

```
def setup_tabs(self):
    self.tab_widget = QTabWidget(self)
    self.setCentralWidget(self.tab_widget)

    self.info_tab = QWidget()
    self.ui_info = Ui_InfoTab()
    self.ui_info.setupUi(self.info_tab, self.tab_widget)
    self.tab_widget.addTab(self.info_tab, "Інформація")

    self.fence_tab = Ui_FenceTab()
    self.tab_widget.addTab(self.fence_tab, "Додати конструкції")

    self.summary_tab_widget = QWidget()
    self.ui_summary = Ui_SummaryTab()
    self.ui_summary.setupUi(self.summary_tab_widget)
    self.tab_widget.addTab(self.summary_tab_widget, "Підсумок")
```

Рис. 2.13. Код методу `setup_tabs`

Метод `setup_tabs` відповідає за створення вкладок в основному вікні програми. Спочатку створюється об'єкт `QTabWidget`, який буде містити вкладки. Цей об'єкт встановлюється як центральний віджет вікна за допомогою методу `setCentralWidget`.

Далі створюються три вкладки:

1. Вкладка "Інформація" – для цієї вкладки створюється об'єкт `QWidget`, на який налаштовується інтерфейс через `Ui_InfoTab`, який відповідає за відображення інформації про додаток.
2. Вкладка "Додати конструкції" – для цієї вкладки використовується об'єкт `Ui_FenceTab`, що відповідає за додавання конструкцій будівлі.
3. Вкладка "Підсумок" – створюється ще одна вкладка для відображення підсумкових результатів розрахунків, на якій відображається відповідний інтерфейс через `Ui_SummaryTab`.

2.3.4. Реалізація вкладки з інформацією про додаток

Вкладка "Інформація" є частиною інтерфейсу програми і відображає користувачу загальну інформацію про додаток, а також надає можливість переходу до вкладки для додавання конструкцій. Вкладка створюється за допомогою класу `Ui_InfoTab`, який відповідає за розміщення елементів на вкладці та їхній вигляд.

```
class Ui_InfoTab:
    def setupUi(self, InfoTab: QWidget, tab_widget):
        layout = QVBoxLayout()
        layout.setContentsMargins(20, 20, 20, 20)
        layout.setSpacing(20)

        title_label = QLabel("Інформація про програму", InfoTab)

        title_label.setAlignment(Qt.AlignCenter)
        layout.addWidget(title_label)

        info_text = QLabel(
            "Це будівельний калькулятор для розрахунку профілю теплового навантаження будівлі.\n\n"
            "1. Для додавання огорожувальної конструкції перейдіть на вкладку 'Додати конструкції'.\n"
            "2. Після введення всіх даних перейдіть на вкладку 'Підсумок' для отримання розрахунків та підсумкового файлу.",
            InfoTab
        )
        info_text.setStyleSheet(
            "font-size: 18px; "
            "color: #555; "
            "line-height: 1.6; "
            "background-color: #fafafa; "
            "border-radius: 5px; "
            "padding: 20px;"
        )
        info_text.setWordWrap(True)
        layout.addWidget(info_text)

        button = QPushButton("Перейти до 'Додати конструкції'", InfoTab)

        button.clicked.connect(lambda: tab_widget.setCurrentIndex(1))

        layout.addWidget(button)

        spacer = QSpacerItem(20, 40, QSizePolicy.Minimum, QSizePolicy.Expanding)
        layout.addItem(spacer)

        InfoTab.setLayout(layout)
```

Рис. 2.14. Клас `Ui_InfoTab`

Клас `Ui_InfoTab` містить метод `setupUi`, який відповідає за ініціалізацію та розміщення всіх елементів на вкладці. Для організації розміщення елементів використовується вертикальний лейаут (`QVBoxLayout`) [40]. Верхня частина вкладки містить мітку (`QLabel`) [37] з текстом "Інформація про програму". Нижче заголовка розміщується ще одна мітка, яка містить текст з описом призначення програми та інструкцією щодо подальших дій. Цей текст надається користувачеві у вигляді `QLabel` з автоматичним перериванням рядків.

Під текстом розміщується кнопка (QPushButton) [33], яка дає можливість користувачу перейти до вкладки "Додати конструкції". Для цього використовується сигнал clicked, який підключається до методу setCurrentIndex.

Результат відкриття цієї вкладки наведено на Рис. 2.15.

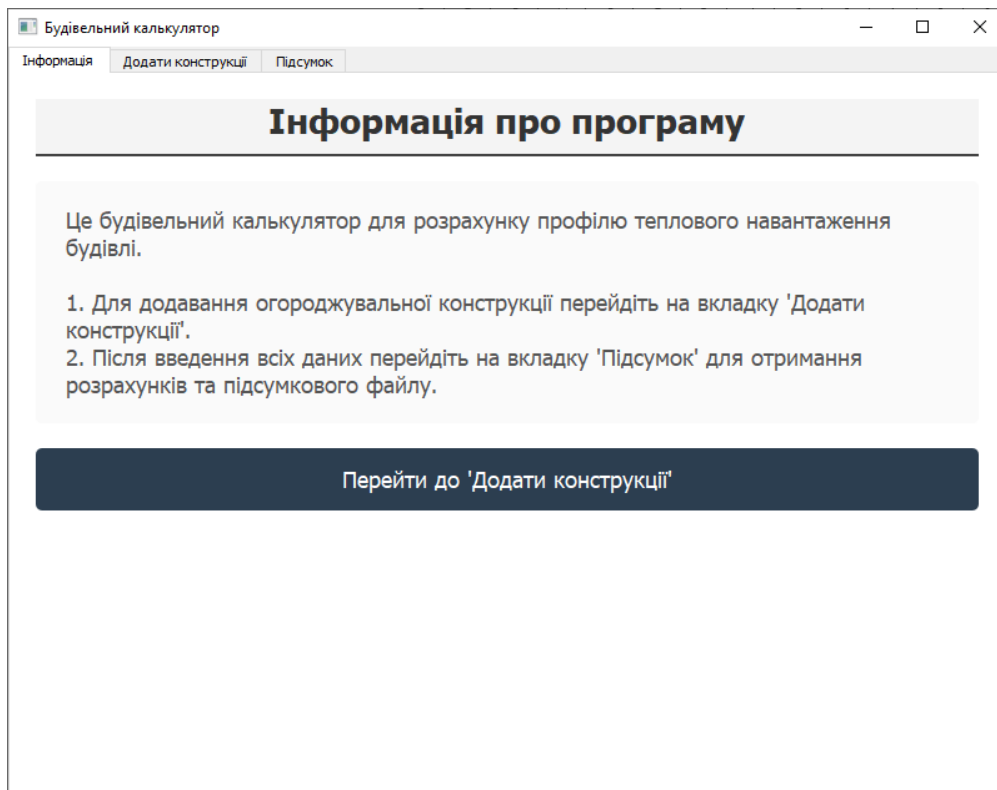


Рис. 2.15. Вигляд вкладки «Інформація»

Як відображено на Рис. 2.6, у результаті відкрилася перша вкладка програми із зазначеною інформацією про призначення додатка та короткою інструкцією користування.

2.3.5. Реалізація вкладки для додавання конструкцій

Вкладка для додавання конструкцій реалізована у файлі ui/fence_ui.py, код цього файлу наведено у додатку А.1.

Файл fence_ui.py містить основний функціонал для вкладки програми, що забезпечує взаємодію користувача з будівельними конструкціями огорожень та їх рівнями. У рамках цього класу реалізовано створення, редагування, збереження, завантаження та видалення конструкцій та рівнів. Весь функціонал

побудований навколо графічного інтерфейсу, що використовує компоненти бібліотеки PyQt5.

Клас `Ui_FenceTab` ініціалізує графічний інтерфейс через метод `init_ui`. На основі компонента `QVBoxLayout` створюється вертикальне розміщення елементів на вкладці. Створюється кнопка для додавання нових конструкцій, яка при натисканні відкриває діалогове вікно для введення параметрів нової конструкції. Окрім цього, на вкладці є кнопка для очищення всіх введених даних. Після очищення даних на вкладці відображатиметься оновлений список конструкцій і рівнів.

Використано `QScrollArea` [39], що дозволяє динамічно прокручувати елементи вкладки. У даному випадку список конструкцій та рівнів будівлі додається до цієї області, що дозволяє обмежити використання простору на екрані та забезпечити користувачеві можливість прокручування за необхідності. Це досягається через використання `QFrame` [36], на якому реалізовано вертикальне розміщення кнопок для кожної конструкції та її рівнів.

При натисканні кнопки для створення конструкції викликається метод `show_fence_dialog`. Це діалогове вікно дозволяє користувачу ввести необхідні дані для нової конструкції або відредагувати існуючу. У діалоговому вікні передбачені поля для введення площі конструкції, внутрішньої та зовнішньої температури, а також вибору типу конструкції. Тип конструкції вибирається з комбінованого списку, який заповнюється через функцію `get_enclosure_types`, що отримує доступні типи конструкцій.

Після введення всіх параметрів користувач може зберегти конструкцію за допомогою кнопки "Зберегти". Дані про конструкцію передаються в метод `save_fence`, який здійснює перевірку введених значень. Перш за все перевіряється, чи заповнені всі необхідні поля і чи є введені значення числами, що відповідають вимогам. У разі, якщо дані не відповідають формату, користувачеві виводиться повідомлення про помилку.

При збереженні нової конструкції або редагуванні існуючої конструкції здійснюється перевірка коректності рядка з типом конструкції. Якщо рядок має

правильний формат, дані конструкції зберігаються в об'єкті будівлі, який являє собою інстанс класу `Building`. Якщо тип конструкції має параметри в дужках, то вони розбираються і додаються до відповідних полів конструкції.

Для кожної конструкції також можна додавати рівні, кожен з яких має свої властивості: матеріал, теплопровідність та товщина. Кожен рівень також можна редагувати або видаляти через відповідні діалогові вікна. Для вибору матеріалу використовується комбінований список, що заповнюється через функцію `get_materials`. Введення товщини рівня здійснюється через поле для тексту. Подібно до створення та редагування конструкції, для кожного рівня передбачено перевірку правильності введених даних, а також можливість їх збереження або видалення.

На вкладці реалізовано динамічне створення кнопок для кожної конструкції та кожного рівня. Ці кнопки з'являються в області прокручування, що дозволяє користувачеві взаємодіяти з елементами будівлі після їх додавання або редагування. Для кожної конструкції створюється кнопка, що містить основну інформацію про конструкцію, таку як площа, температури та коефіцієнт. При натисканні на кнопку відкривається відповідне діалогове вікно для редагування або перегляду деталей конструкції.

Аналогічно, для кожного рівня створюється окрема кнопка, яка містить основні параметри рівня: матеріал, теплопровідність та товщина. Кнопка також містить функціональність для редагування рівня, а також додавання нового рівня.

Збереження даних здійснюється через функцію `save_building_data`, яка використовує функцію `save_data_fence` для запису об'єкта будівлі в файл JSON. Це дозволяє зберігати всі конструкції та рівні на диску і відновлювати їх при наступному запуску програми. Для завантаження даних використовується метод `load_building_data`, який викликає функцію `load_data_fence`. Вона зчитує дані з файлу JSON і відновлює їх у програмі. При цьому після завантаження даних оновлюється список конструкцій та рівнів на вкладці.

Якщо при збереженні або завантаженні даних виникають помилки (наприклад, через неправильний формат файлу або відсутність доступу до файлу), користувач отримує відповідне повідомлення через діалогове вікно з помилкою.

Кнопка "Очистити всі дані" дозволяє видалити всі додані конструкції та рівні з об'єкта будівлі. Після очищення даних викликається метод `update_scroll_area`, який оновлює відображення на вкладці, видаляючи всі кнопки для конструкцій і рівнів. Користувач отримує повідомлення про успішне очищення даних через діалогове вікно.

Об'єктом для зберігання даних є клас `Building`, який містить список всіх конструкцій типу `Fence`. Кожна конструкція має власний список рівнів типу `Level`. Кожен рівень включає матеріал, що представлений класом `Material`, і має параметри товщини та теплопровідності. Для кожної конструкції та рівня існують відповідні методи додавання, редагування та видалення.

Вигляд цієї вкладки з доданими конструкціями наведено на рис. 2.16.

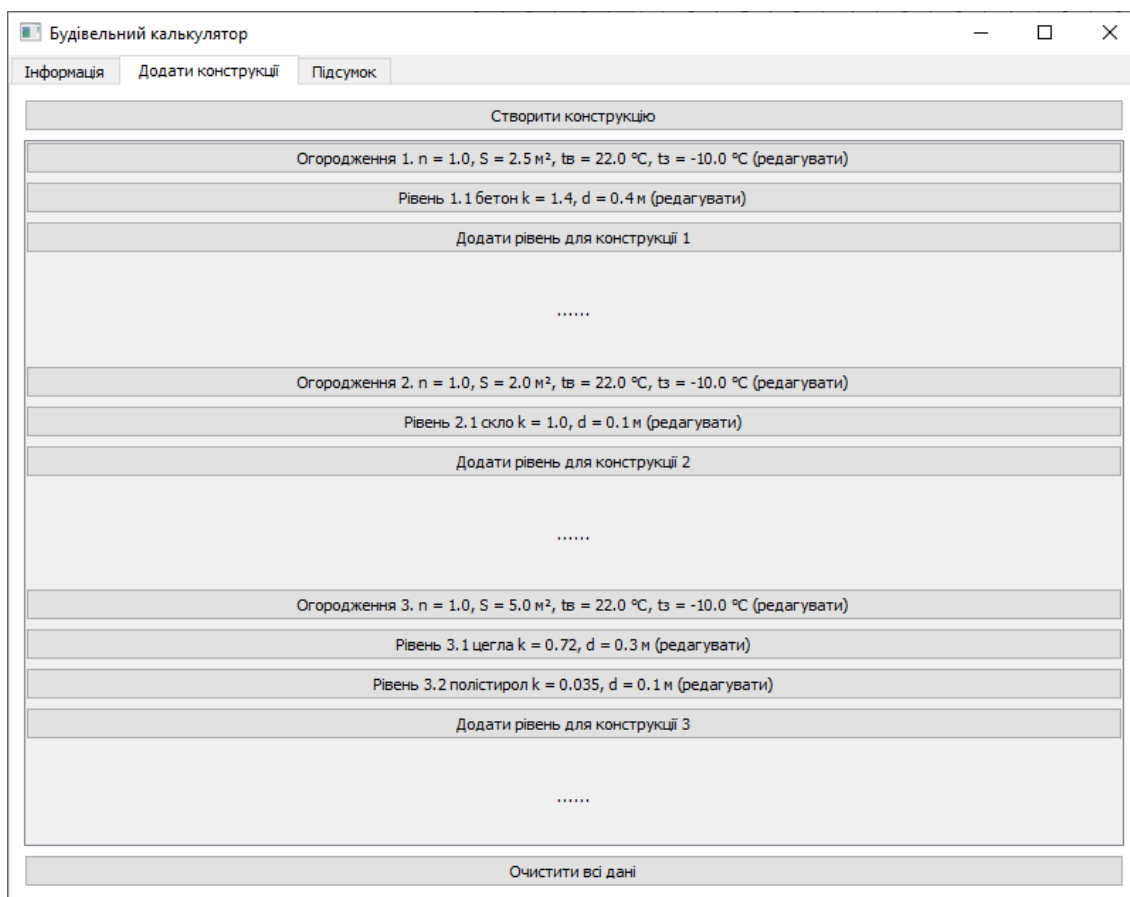


Рис. 2.16. Вигляд вкладки «Додати конструкції»

2.3.6. Реалізація вкладки для отримання результатів розрахунків

У вкладці для отримання результатів розрахунків реалізовано інтерфейс, що дозволяє користувачу вводити параметри для визначення теплового навантаження конструкцій будівлі, виконувати відповідні обчислення та генерувати звіт.

Клас `Ui_SummaryTab`, що реалізує вкладку результатів розрахунків, створюється через метод `setupUi`. Код цього методу наведено на Рис. 2.17. Він включає ініціалізацію всіх елементів інтерфейсу, таких як комбінований список для вибору віку будівлі, текстове поле для введення року побудови, кнопка для запуску розрахунків та таблиця для відображення результатів.

```
class Ui_SummaryTab:
    def setupUi(self, SummaryTab):
        layout = QVBoxLayout()
        layout.setSpacing(20)
        layout.setContentsMargins(20, 20, 20, 20)

        age_label = QLabel("Виберіть  введіть вік будівлі:")
        age_label.setStyleSheet("font-size: 14px;")
        layout.addWidget(age_label)

        self.age_selector = QComboBox()
        self.age_selector.addItem("До 1985")
        self.age_selector.addItem("Після 1985")
        layout.addWidget(self.age_selector)

        self.year_input = QLineEdit()
        self.year_input.setPlaceholderText("Введіть рік побудови (наприклад, 1980)")
        layout.addWidget(self.year_input)

        calculate_button = QPushButton("Розрахувати")
        calculate_button.setStyleSheet("font-size: 14px; font-weight: bold;")
        calculate_button.clicked.connect(self.perform_calculation)
        layout.addWidget(calculate_button)

        self.total_label = QLabel("Загальна теплова потужність: -")
        self.total_label.setStyleSheet("font-size: 18px; font-weight: bold; color: #2C3E50;")
        layout.addWidget(self.total_label)

        self.construction_label = QLabel("Потужність від конструкцій: -")
        layout.addWidget(self.construction_label)

        self.ventilation_label = QLabel("Потужність від вентиляції: -")
        layout.addWidget(self.ventilation_label)

        self.table = QTableWidgetItem()
        self.table.setColumnCount(2)
        self.table.setHorizontalHeaderLabels(["Тип конструкції", "Q конструкції (Вт)"])
        self.table.setStyleSheet("font-size: 14px;")
        layout.addWidget(self.table)

        header = self.table.horizontalHeader()
        header.setSectionResizeMode(0, QHeaderView.Stretch)
        header.setSectionResizeMode(1, QHeaderView.Stretch)

        download_report_button = QPushButton("Завантажити звіт")
        download_report_button.setStyleSheet("font-size: 14px; font-weight: bold; color: #FFFFFF; background-color: #3498DB;")
        download_report_button.clicked.connect(self.generate_report)
        layout.addWidget(download_report_button)

        SummaryTab.setLayout(layout)
```

Рис. 2.17. Код для створення класу `Ui_SummaryTab`

Основний функціонал вкладки передбачає наступне. Користувач може вибрати варіант віку будівлі або ввести рік побудови вручну. Для цього передбачено два основні елементи: комбінований список QComboBox [35], що дозволяє вибрати рік будівлі (до 1985 чи після 1985), та текстове поле QLineEdit [31], яке дозволяє ввести конкретний рік будівлі. При натисканні кнопки "Розрахувати" викликається метод `perform_calculation`, що здійснює обчислення на основі наданих даних.

Метод `perform_calculation` (код наведено на Рис. 2.18) отримує значення з комбінованого списку та текстового поля для визначення року будівлі. Якщо рік введено вручну, він перевіряється на коректність. Якщо рік некоректний або не введено значення, застосовуються стандартні значення: 1985 для старих будівель та поточний рік для нових. Після цього викликається функція `calculate_heat_load`, що виконує обчислення на основі даних про будівлю.

```
def perform_calculation(self):
    selected_age = self.age_selector.currentText()
    input_year = self.year_input.text()

    if input_year.isdigit():
        self.building_year = int(input_year)
    elif selected_age == "До 1985":
        self.building_year = 1985
    else:
        self.building_year = 2024

    file_path = "json_data/building_data_fence.json"
    self.results = calculate_heat_load(file_path, self.building_year)

    self.total_label.setText(f"Загальна теплова потужність: {self.results['total_load']:.2f} Вт")
    self.construction_label.setText(f"Потужність від конструкцій: {self.results['construction_load']:.2f} Вт")
    self.ventilation_label.setText(f"Потужність від вентиляції: {self.results['ventilation_load']:.2f} Вт")

    self.table.setRowCount(len(self.results['construction_details']))
    for row, detail in enumerate(self.results['construction_details']):
        self.table.setItem(row, 0, QTableWidgetItem(detail['type']))
        self.table.setItem(row, 1, QTableWidgetItem(f"{detail['q_load']:.2f}"))
```

Рис. 2.18. Код методу `perform_calculation`

Результати розрахунків зберігаються в змінній `self.results`, після чого оновлюються відображення на екрані: загальна теплове навантаження,

потужність від конструкцій та вентиляції, а також деталі конструкцій будівлі. Для відображення результатів використовується таблиця `QTableWidget` [32], у яку додаються рядки з даними про конструкції та їх теплові навантаження.

Вигляд вкладки з результатами розрахунків наведено на Рис. 2.19.

Будівельний калькулятор

Інформація Додати конструкції Підсумок

Виберіть або введіть рік будівлі:

До 1985

2024

Розрахувати

Загальна теплова потужність: 450.83 Вт

Потужність від конструкцій: 281.77 Вт

Потужність від вентиляції: 169.06 Вт

	Тип конструкції	Q конструкції (Вт)
1	Зовнішні стіни, покриття з штучними матеріалами і ...	180.13
2	Перекриття над технічними підпіллями нижче рівня ...	67.08
3	Перекриття над неопалювальними підвалами з ...	34.56

Завантажити звіт

Рис. 2.19. Вигляд вкладки «Підсумок»

Також передбачена кнопка для генерації звіту, яка викликає метод `generate_report`. Цей метод описано в додатку А.2. Він генерує PDF-звіт на основі отриманих результатів розрахунків за допомогою Python бібліотеки `FPDF2` [26], що включає всю необхідну інформацію про теплове навантаження конструкцій будівлі, а також дані щодо матеріалів та температурних режимів. Звіт зберігається в обраному користувачем файлі у форматі PDF. Вигляд файлу звіту наведено на Рис. 2.20.

Детальний звіт розрахунків	
<i>Рік будівлі: 2024</i>	
<i>Загальна теплова потужність: 450.83 Вт</i>	
<i>Потужність від конструкцій: 281.77 Вт</i>	
<i>Потужність від вентиляції: 169.06 Вт</i>	
Деталі конструкцій:	
1. Тип конструкції: Зовнішні стіни, покриття з штучними матеріалами і вентилязовані перекриття (без захисних стінок).	
Площа: 2.5 м²	
Внутрішня температура: 22.0°C	
Зовнішня температура: -10.0°C	
1.1 Матеріал: бетон, Товщина: 0.4 м	
2. Тип конструкції: Перекриття над технічними підпіллями нижче рівня землі.	
Площа: 2.0 м²	
Внутрішня температура: 22.0°C	
Зовнішня температура: -10.0°C	
2.1 Матеріал: скло, Товщина: 0.1 м	
3. Тип конструкції: Перекриття над неопалювальними підвалами з світловими отворами.	
Площа: 5.0 м²	
Внутрішня температура: 22.0°C	
Зовнішня температура: -10.0°C	
3.1 Матеріал: цегла, Товщина: 0.3 м	
3.2 Матеріал: полістирол, Товщина: 0.1 м	

Рис. 2.20. Вигляд згенерованого файлу зі звітом

У цьому звіті представлена детальна інформація про будівлю, для якої проводилися розрахунки, загальні теплові витрати, окремо теплові витрати на огорожувальні конструкції та вентиляцію. Крім того, наведено деталі щодо кожної конструкції, які були введені користувачем, включаючи їхні матеріали, теплотехнічні характеристики та розрахункові температурні режими.

РОЗДІЛ 3. ПЕРСПЕКТИВИ ВДОСКОНАЛЕННЯ ПІДХОДІВ ДО РОЗРАХУНКУ ТЕПЛОВОГО НАВАНТАЖЕННЯ БУДІВЕЛЬ

3.1. Пропозиції щодо розвитку застосунку

У зв'язку з розвитком технологій та зростаючими вимогами до енергозбереження, розробка та вдосконалення програмного забезпечення для розрахунку теплового навантаження будівель є необхідною складовою частиною проектування систем опалення. Пошук оптимальних рішень у цьому напрямі потребує постійної уваги до удосконалення існуючих методів та інструментів, зокрема програмних застосунків, які реалізують ці розрахунки. Виходячи з аналізу наявних підходів та результатів розрахунків, можна виділити кілька напрямів для вдосконалення функціональності програмного забезпечення в частині розрахунку теплового навантаження.

Одним із перспективних напрямків розвитку є інтеграція застосунку з базами даних, що містять інформацію про будівельні матеріали, конструкції та типи інженерних систем. Це дозволить значно полегшити процес введення даних, а також знизити ймовірність помилок у розрахунках через невірно введену інформацію. Оскільки більшість таких застосунків використовують бази даних для зберігання технічних характеристик матеріалів, розширення та оновлення цієї бази стане значним кроком вперед. Для цього доцільно залучити виробників будівельних матеріалів і постачальників інженерних систем до процесу наповнення бази актуальними даними, що дозволить програмі автоматично обирати оптимальні параметри конструкцій залежно від конкретних умов.

Додатковим кроком може стати розширення функціоналу застосунку, що дасть змогу здійснювати розрахунки теплового навантаження з урахуванням реальних кліматичних умов. Для цього в програму варто впровадити можливість інтеграції з базами кліматичних даних, що дозволить користувачеві обирати необхідні значення температур та інших кліматичних характеристик

для розрахунків без необхідності вручну вводити ці дані. Це також може включати автоматичну корекцію температурних режимів залежно від конкретного сезону року або погодних змін у реальному часі, що дозволить значно підвищити точність результатів розрахунків.

Застосунок має бути здатним враховувати не лише характеристики зовнішніх факторів, але й внутрішні процеси в будівлі, такі як теплові потоки від побутових приладів, освітлення та людська діяльність. Врахування цих параметрів може мати значення для точного визначення потреби в енергії для підтримання комфортної температури. Враховуючи розвиток технологій інтернету речей (IoT), варто передбачити можливість підключення датчиків температури та вологості, що дозволяє здійснювати динамічний моніторинг і автоматичне коригування параметрів системи опалення, а саме код, що наведено у файлі додатку A.1.

Ще одним напрямом вдосконалення є інтерфейс користувача (UI/UX), що значно спрощує взаємодію з програмою. Це дозволить знизити час на навчання користувачів та зменшити ймовірність помилок при введенні даних. Створення інтуїтивно зрозумілих інтерфейсів, що підтримують різноманітні формати вводу та виводу даних, полегшить використання застосунку як для досвідчених фахівців, так і для початківців у галузі проектування систем опалення.

Також варто звернути увагу на інтеграцію з іншими програмними засобами та системами управління будівельними проектами. Завдяки такій інтеграції можна автоматизувати обмін даними між різними етапами проектування та реалізації, що дозволить уникнути необхідності повторного введення однакових даних в різних програмних продуктах. Це має особливе значення при роботі з великими проектами, де застосовуються різні програмні продукти для розрахунку різних аспектів будівлі та її інженерних систем.

У контексті розвитку застосунку також варто звернути увагу на покращення точності розрахунків. Це може бути досягнуто шляхом застосування новітніх методів моделювання, що враховують більш детально теплопередачу через конструкції, а також динамічні зміни температури в межах

окремих приміщень та їх взаємодії з системами вентиляції та кондиціонування. Врахування детальніших фізичних процесів дозволить отримати точніші результати, що, у свою чергу, дозволить краще планувати ресурси для опалення.

Зважаючи на те, що енергозбереження є однією з пріоритетних задач у будівельній галузі, варто також звернути увагу на можливість виконання розрахунків з урахуванням альтернативних джерел енергії. Зокрема, це може стосуватися сонячних панелей, теплових насосів або інших джерел енергії, які можуть бути використані для часткового або повного забезпечення будівлі енергією для опалення. Інтеграція таких джерел в застосунок дозволить здійснювати більш комплексний аналіз енергоспоживання та заощаджень на опаленні.

Для подальшого розвитку застосунку, варто також розглянути можливість інтеграції з системами автоматизованого управління будівлею (BMS), що дозволить здійснювати моніторинг і коригування роботи системи опалення в реальному часі на основі отриманих даних про температурні режими та потреби в енергії. Це дозволить досягти значної економії енергії в реальному часі, що особливо вигідно для великих будівель та комплексів, де можливість оптимізувати витрати на опалення може забезпечити додаткові фінансові заощадження.

3.2. Рекомендації щодо впровадження розробленого застосунку та його значимість

Впровадження розробленого застосунку для розрахунку теплового навантаження будівель вимагає уважного підходу до кількох аспектів, що впливають на успіх цього процесу. Перш за все, потрібно забезпечити сумісність програмного забезпечення з іншими системами, що використовуються в будівельній галузі. Це дозволить знизити час, необхідний для інтеграції програми в існуючі робочі процеси, а також сприятиме

безперешкодному обміну даними між різними етапами проектування та реалізації будівельних проектів. Розширення функціональних можливостей застосунку шляхом інтеграції з іншими системами й базами даних дозволить підвищити точність розрахунків та зменшити ймовірність помилок при введенні даних.

Наступним аспектом є адаптація застосунку до різних типів будівель. Для цього доцільно забезпечити можливість вибору різних типів конструкцій та інженерних систем із бази даних, яка регулярно оновлюється відповідно до нових технічних рішень та нормативних вимог.

У процесі впровадження треба передбачити навчання користувачів. Оскільки застосунок може використовуватися як досвідченими фахівцями, так і новачками в галузі, необхідно створити навчальні матеріали та курси, які допоможуть користувачам освоїти роботу з програмою. Інтерфейс має бути зрозумілим і зручним, що значно зменшує час на навчання та знижує ймовірність помилок при введенні даних. Крім того, забезпечення наявності технічної підтримки є необхідним для оперативного вирішення проблем і допомоги користувачам.

Оскільки будівельні матеріали та технології постійно змінюються, а також змінюються нормативні вимоги, необхідно регулярно оновлювати базу даних програми. Це дасть змогу підтримувати точність розрахунків та відповідність новітнім стандартам. Система оновлення має бути автоматизованою, що дозволить постійно мати актуальні дані без потреби ручного втручання.

Враховуючи складність проектування систем опалення для великих об'єктів, впровадження функцій для спільної роботи над проектом стане вигідним. Це дозволить кільком користувачам одночасно працювати над одним проектом, зберігаючи всі дані в реальному часі. Впровадження цієї функції через хмарне сховище сприятиме поліпшенню комунікації і прискорить процес розробки проекту.

Інтеграція системи перевірки даних дозволить підвищити точність розрахунків і знизити ймовірність помилок у введених параметрах. Автоматичні системи контролю, які порівнюють введені значення з нормативними даними, дозволять своєчасно виявляти можливі невідповідності. Це дасть змогу на кожному етапі проектування впевнено працювати з точними даними.

Розширення можливостей застосунок для врахування додаткових параметрів, таких як інсоляція, вентиляція та інші аспекти, дозволить зробити розрахунки більш точними. Це також допоможе здійснювати детальний аналіз енергоспоживання будівлі та оптимізувати планування системи опалення, що забезпечить зменшення витрат на енергію що буде відображатись у кінцевому звіті в рамках розширення коду цього файлу, що наведено у додатку А.2.

Розроблений застосунок також сприятиме підвищенню енергетичної ефективності будівель. Точніші розрахунки теплового навантаження дозволять оптимізувати роботу системи опалення, знижуючи її енерговитрати та забезпечуючи комфортні умови для мешканців чи користувачів будівлі. Врахування різних факторів, таких як теплотехнічні характеристики матеріалів, кліматичні умови та специфіка інженерних систем, дозволить створювати систему опалення, що працює з мінімальними витратами енергії.

Впровадження цього інструменту стане частиною процесу проектування та експлуатації будівель, що дозволить значно знизити енергоспоживання та покращити енергетичну ефективність будівель. Застосування системи автоматизує процеси проектування та розрахунків, зменшуючи людський фактор та підвищуючи точність результатів. Це дозволить спростити і пришвидшити процес проектування, а також досягти значної економії енергоресурсів.

ВИСНОВКИ

В межах виконаної кваліфікаційної роботи було розглянуто проблему автоматизації розрахунку теплового навантаження будівель. Оскільки теплові втрати будівель є однією з основних причин високого енергоспоживання, забезпечення точних і швидких розрахунків теплового навантаження є умовою для розробки та впровадження енергоефективних рішень у будівництві. У рамках роботи було поставлено завдання розробити програмне забезпечення, яке дозволить автоматизувати процес оцінки теплових втрат будівель, враховуючи різні фактори, такі як конструктивні елементи, кліматичні умови, а також вплив внутрішніх джерел тепла.

У процесі дослідження було проаналізовано математичні моделі тепловтрат, а також визначено найбільш оптимальні методи для впровадження в програмне забезпечення. Аналіз існуючих підходів до обчислення теплових втрат показав необхідність інтеграції цих моделей у зручний інтерфейс, який би забезпечував простоту використання та точність результатів для практичних застосувань.

Для досягнення поставленої мети було розроблено алгоритми, що дозволяють розраховувати теплові втрати через основні конструктивні елементи будівлі, включаючи стіни, вікна, двері, дах і підлогу. Для цього використано фізичні характеристики матеріалів, а також кліматичні умови, що дозволяє отримувати точні результати для конкретних будівель. Врахування різних факторів, таких як різниця температур між внутрішнім і зовнішнім середовищем, тип матеріалу конструкцій, їх товщина та коефіцієнти теплопередачі, дало змогу побудувати точні моделі теплопередачі. Алгоритми були розроблені з урахуванням нормативних вимог та стандартів у галузі теплотехніки.

З метою забезпечення зручності використання було створено інтерфейс користувача, що дозволяє вводити необхідні дані про характеристики будівлі, такі як розміри конструкцій, типи матеріалів, кількість вікон та дверей, а також

місце розташування будівлі для врахування кліматичних умов. Усі ці параметри зібрані в одному інтерфейсі, що забезпечує швидкість і зручність введення даних користувачем. Крім того, реалізовано механізм для введення даних вручну, що забезпечує більшу гнучкість у використанні інструменту в різних умовах.

Розроблене програмне забезпечення також дозволяє формувати звіт про теплове навантаження будівлі на основі результатів розрахунку теплових втрат. Звіт містить всі необхідні дані про втрати тепла через різні конструктивні елементи будівлі, а також результати для окремих приміщень і всієї будівлі в цілому. Звіт генерується у форматі PDF, що дозволяє зберігати та передавати його для подальшого аналізу або використання у проектуванні системи опалення, вентиляції та кондиціонування.

Науковий підхід до розв'язання поставленої задачі передбачав поетапне виконання кожного етапу дослідження, починаючи з огляду теоретичних основ і методик розрахунків теплових навантажень, і закінчуючи розробкою інтерфейсу користувача та програмного забезпечення для автоматизації цього процесу. Використання мови програмування Python та бібліотек PyQt5 дозволило створити інтерфейс, що забезпечив зручну роботу з даними та високу гнучкість у роботі програмного забезпечення.

Результатом дослідження є розробка програмного інструменту, що дозволяє швидко і точно оцінювати теплове навантаження будівель, що відповідає поставленим завданням. Тестування програми на реальних даних підтвердило її працездатність і точність результатів. Програма апробована на тестових даних і продемонструвала правильність розрахунків і можливість її використання для практичних цілей у галузі проектування енергозберігаючих будівель та систем опалення. Розроблений інструмент може бути використаний інженерами-проектувальниками при плануванні теплопостачання будівель, а також у навчальних цілях для підготовки спеціалістів у сфері енергоефективності та теплоенергетики.

Для подальшого удосконалення програми пропонується розширити базу матеріалів та конструкцій, що використовуються в розрахунках, а також додати можливість імпорту даних з різних файлів, таких як CSV або JSON, що дозволить автоматизувати процес введення даних. Розробка веб-версії програми також є напрямом для збільшення доступності інструменту, що дозволить здійснювати розрахунки з різних пристроїв без необхідності встановлення програмного забезпечення на кожен комп'ютер. Такий підхід забезпечить зручність і гнучкість у використанні програми для професійних і навчальних цілей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Боженко М. Ф. Теплові навантаження споживачів : Навчальний посібник. Київ, 2022. 67 с.
2. Визначення необхідного повітрообміну приміщень. Державні Будівельні Норми України. URL: https://dbn.co.ua/publ/viznachennja_neobkhdno_go_povitroobminu_primishhen/35-1-0-1212 (дата звернення: 15.12.2024).
3. Гільчук А. В. Теорія теплопровідності : Підручник. Київ, 2022.
4. ДБН В.2.6- 31 2006. Конструкції будинків і споруд. Теплова ізоляція будівель.
5. Долгополов І. Визначення теплового навантаження будівель та вибір системи тепlopостачання : Навчальний посібник. Київ, 2011. 119 с.
6. ДСТУ Б А.2.2-12:2015. Енергоефективність будівель. Метод розрахунку енергоспоживання для опалення, охолодження, вентиляції, освітлення та гарячого водопостачання К.: Мінрегіон України, 2015. 205 с.
7. Зосім М. Діаграми потоків даних (Data Flow Diagrams). Махум Zosym. URL: <https://www.maxzosim.com/data-flow-diagrams/> (дата звернення: 15.12.2024).
8. Кузнецова Я. Ю. Використання екологічних стандартів в проектуванні залізничних вокзальних комплексів. Сучасні проблеми архітектури та містобудування : наук.-техн. зб. / відп. ред. М. М. Дьомін. Київ : КНУБА, 2014. Випуск № 36. С. 110–116.
9. Методи JSON, toJSON. Сучасний підручник з JavaScript. URL: <https://uk.javascript.info/json> (дата звернення: 15.12.2024).
10. Методологія ideo. Stud. URL: https://stud.com.ua/87184/ekonomika/metodologiya_ideo (дата звернення: 15.12.2024).
11. Могилевич Д. Інструментальні засоби python для моделювання та системного аналізу часових рядів при вирішенні задач кіберзахисту інформаційно-комунікаційних систем : Навчальний посібник. Київ, 2023. с. 14 – 15.

12. Навчальний посібник із PyQt5 із прикладами: проектування графічного інтерфейсу за допомогою PyQt Python. Guru99. URL: <https://www.guru99.com/uk/pyqt-tutorial.html> (дата звернення: 15.12.2024).
13. Пономарчук І. А. Опалення : Навчальний посібник. Вінниця, 2017. 130 с.
14. Попирев Д. О. Огляд продукції компанії-розробника JetBrains. Free and open source software : XIII міжнар. науково-практ. конф., м. Харків, 16–18 листоп. 2021 р. Харків, 2021.
15. Проценко О. Р. Ізоляція електротехнічного обладнання : Курс лекцій. Київ, 2022.
16. Розрахунок теплового навантаження будівлі. OTOS. URL: https://www.otos.com.ua/rozrahunok_teplovogo_navantagennya (дата звернення: 15.12.2024).
17. Савастюк С. В. Технології розроблення програмного забезпечення : Підручник. Київ, 2022.
18. Садиков М. Рівні мов програмування. JavaRush. URL: <https://javarush.com/ua/groups/posts/uk.1904.rvn-mov-programuvannja> (дата звернення: 15.12.2024).
19. Системи та схеми вентиляції. Проектний розрахунок систем вентиляції. Elib LNTU. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/Синій%201/page9.html (дата звернення: 15.12.2024).
20. Слюсаренко Ю. Методи вибору теплоізоляційного матеріалу для утеплення будівель : посібник для проектування. Київ, 2014.
21. Хлицов М. Енергозбереження в будівельній індустрії : Методичні вказівки. Одеса, 2017.
22. Шовкалюк М. Аналіз тепловтрат через огороження з урахуванням різних методів оцінки теплозахисних властивостей. Енергетика: економіка, технології, екологія : №4, 2017 р.
23. Що таке об'єктно-орієнтоване програмування: принципи, переваги та недоліки. GoIT. URL: <https://goit.global/ua/articles/shcho-take-ob-iektno->

orrientovane-prohramuvannia-pryntsypu-perevahy-ta-nedoliky/ (дата звернення: 12.12.2024).

24. About draw.io. draw.io. URL: <https://www.drawio.com/about> (дата звернення: 15.12.2024).

25. Difference Between Matplotlib VS Seaborn. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/difference-between-matplotlib-vs-seaborn/> (дата звернення: 15.12.2024).

26. fpdf2 // GitHub Pages. URL: <https://alexanderankin.github.io/pyfpdf/>

27. json.load() in Python. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/json-load-in-python/> (дата звернення: 15.12.2024).

28. Kattamuri M. Introduction to Matplotlib - GeeksforGeeks. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/python-introduction-matplotlib/> (дата звернення: 12.12.2024).

29. NumPy Introduction & Guide for Beginners. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/introduction-to-numpy/> (дата звернення: 11.12.2024).

30. PyQt in Python. Medium. URL: <https://medium.com/@krishnusai/building-cross-platform-gui-applications-with-pyqt-a-comprehensive-guide-f0550bef8221> (дата звернення: 03.12.2024).

31. PyQt5 - QLineEdit. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/pyqt5-qlineedit/> (дата звернення: 09.12.2024).

32. PyQt5 - QTableWidgetItem. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/pyqt5-qtablewidget/> (дата звернення: 09.12.2024).

33. PyQt5 - QPushButton. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/pyqt5-qpushbutton/> (дата звернення: 09.12.2024).

34. Python interface to Tcl/Tk. Python documentation. URL: <https://docs.python.org/uk/3/library/tkinter.html> (дата звернення: 09.12.2024).

35. QComboBox - Qt for Python. Qt Documentation. URL: <https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QComboBox.html> (дата звернення: 09.12.2024).

36. QFrame - Qt for Python. Qt Documentation. URL: <https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QFrame.html> (дата звернення: 09.12.2024).
37. QLabel - Qt for Python. Qt Documentation. URL: <https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QLabel.html> (дата звернення: 09.12.2024).
38. QMainWindow - Qt for Python. Qt Documentation. URL: <https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QMainWindow.html> (дата звернення: 09.12.2024).
39. QScrollArea - Qt for Python. Qt Documentation. URL: <https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QScrollArea.html> (дата звернення: 09.12.2024).
40. QVBoxLayout - Qt for Python. Qt Documentation. URL: <https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QVBoxLayout.html> (дата звернення: 09.12.2024).

ДОДАТКИ

Додаток А

Текст коду програми

A.1 – Код файлу ui\fence_ui.py

```

from PyQt5.QtWidgets import QWidget, QVBoxLayout,
QPushButton, QScrollArea, QFrame, QLabel, QDialog, QComboBox,
QLineEdit, QFormLayout
from PyQt5.QtCore import Qt

from json_data.json_utils import get_enclosure_types,
get_materials, save_data_fence, load_data_fence
from model.classes import Fence, Level, Material, Building

def is_valid_number(value: str) -> bool:
    try:
        float(value)
        return True
    except ValueError:
        return False

class Ui_FenceTab(QWidget):
    def __init__(self):
        super().__init__()
        self.init_ui()
        self.building = Building()
        self.filename = "json_data/building_data_fence.json"
        self.load_building_data()

    def init_ui(self):
        self.layout = QVBoxLayout(self)

        self.create_fence_button = QPushButton("Створити
конструкцію", self)

self.create_fence_button.clicked.connect(self.on_create_fence)

        self.layout.addWidget(self.create_fence_button)

        self.scroll_area = QScrollArea(self)
        self.scroll_area.setWidgetResizable(True)

        scroll_widget = QFrame()
        self.scroll_layout = QVBoxLayout(scroll_widget)

        self.scroll_area.setWidget(scroll_widget)
        self.layout.addWidget(self.scroll_area)

        self.scroll_widget = scroll_widget

```

```

        self.scroll_layout.setContentsMargins(0, 0, 0, 0)
        self.scroll_layout.setSpacing(5)

        self.clear_button = QPushButton("Очистити всі дані",
self)

        self.clear_button.clicked.connect(self.clear_data)
        self.layout.addWidget(self.clear_button)

    def on_create_fence(self):
        self.show_fence_dialog()

    def show_fence_dialog(self, fence=None):
        dialog = QDialog(self)
        dialog.setWindowTitle(f"Конструкція
{len(self.building.fences) + 1}" if fence is None else
f"Редагування конструкції {fence.fence_type}")

        form_layout = QFormLayout(dialog)

        enclosure_types = get_enclosure_types()

        type_combo = QComboBox(dialog)
        type_combo.addItems(enclosure_types)

        if fence:
            type_combo.setCurrentText(f"{fence.fence_type}:
{fence.coef}")
            area_input = QLineEdit(dialog)
            area_input.setText(str(fence.area))
            internal_temp_input = QLineEdit(dialog)
internal_temp_input.setText(str(fence.interior_temperature))
            external_temp_input = QLineEdit(dialog)
external_temp_input.setText(str(fence.exterior_temperature))
        else:
            area_input = QLineEdit(dialog)
            internal_temp_input = QLineEdit(dialog)
            external_temp_input = QLineEdit(dialog)

        save_button = QPushButton("Зберегти", dialog)
        save_button.clicked.connect(lambda: self.save_fence(
            fence,
            type_combo.currentText(),
            area_input.text(),
            internal_temp_input.text(),
            external_temp_input.text(),
            dialog
        ))

        delete_button = QPushButton("Видалити", dialog)
        if fence:
            delete_button.clicked.connect(lambda:
self.delete_fence(fence, dialog))

```

```

        form_layout.addRow("Тип конструкції:", type_combo)
        form_layout.addRow("Площа конструкції:", area_input)
        form_layout.addRow("Внутрішня температура:",
internal_temp_input)
        form_layout.addRow("Зовнішня температура:",
external_temp_input)
        form_layout.addRow(save_button)
        form_layout.addRow(delete_button)

        dialog.setLayout(form_layout)
        dialog.exec_()

    def delete_fence(self, fence, dialog):
        self.building.fences.remove(fence)
        self.save_building_data()
        self.update_scroll_area()
        self.print_building_data()
        dialog.accept()

    def save_fence(self, fence, fence_type, area,
internal_temp, external_temp, dialog):
        if not area or not internal_temp or not
external_temp:
            self.show_error("Будь ласка, заповніть всі
поля!")
            return

        if not is_valid_number(area) or not
is_valid_number(internal_temp) or not
is_valid_number(external_temp):
            self.show_error("Будь ласка, введіть коректні
числа у полях площі, внутрішньої та зовнішньої температури.")
            return

        if fence:
            try:
                if "(" in fence_type and ")" in fence_type:
                    fence_name = fence_type.rsplit(" (",
1)[0]

                    params = fence_type.rsplit("(",
1)[1].rstrip(")")

                    coef, alpha_interior, alpha_exterior =
map(float, params.split(","))

                    fence.fence_type = fence_name
                    fence.coef = coef
                    fence.alpha_interior = alpha_interior
                    fence.alpha_exterior = alpha_exterior
                    fence.area = float(area)
                    fence.interior_temperature =
float(internal_temp)

```

```

        fence.exterior_temperature =
float(external_temp)
    else:
        raise ValueError("Неверный формат строки
fence_type")
    except Exception as e:
        print(f"Ошибка при разборе fence_type: {e}")
    else:
        try:
            if "(" in fence_type and ")" in fence_type:
                fence_name = fence_type.rsplit(" (",
1) [0]
                params = fence_type.rsplit("(",
1) [1].rstrip(")")
                coef, alpha_interior, alpha_exterior =
map(float, params.split(", "))
                new_fence = Fence(
                    fence_name,
                    float(external_temp),
                    float(internal_temp),
                    float(area),
                    coef=coef,
                    alpha_interior=alpha_interior,
                    alpha_exterior=alpha_exterior
                )
                self.building.add_fence(new_fence)
            else:
                raise ValueError("Неверный формат строки
fence_type")
        except Exception as e:
            print(f"Ошибка при разборе fence_type: {e}")

    self.save_building_data()
    self.update_scroll_area()
    self.print_building_data()
    dialog.accept()

def update_scroll_area(self):
    for i in reversed(range(self.scroll_layout.count())):
        widget = self.scroll_layout.itemAt(i).widget()
        if widget is not None:
            widget.deleteLater()

    for fence_index, fence in
enumerate(self.building.fences, start=1):
        button_text = f"Огородження {fence_index}. n =
{fence.coef}, S = {fence.area} м², тв =
{fence.interior_temperature} °C, тз = {fence.exterior_temperature}
°C (редагувати)"
        button = QPushButton(button_text)

```

```

        button.clicked.connect(lambda _, fence=fence:
self.show_fence_dialog(fence))
        self.scroll_layout.addWidget(button)

        for level_index, level in enumerate(fence.levels,
start=1):
            level_button_text = f"Рівень
{fence_index}.{level_index} {level.material.material_type} k =
{level.material.thermal_conductivity}, d = {level.thickness} м
(редагувати) "
            level_button = QPushButton(level_button_text)
            level_button.clicked.connect(lambda _,
fence=fence, level=level: self.show_level_dialog(fence, level))
            self.scroll_layout.addWidget(level_button)

            level_button = QPushButton(f"Додати рівень для
конструкції {fence_index}")
            level_button.clicked.connect(lambda _,
fence=fence: self.show_level_dialog(fence))
            self.scroll_layout.addWidget(level_button)

            separator = QLabel(".....")
            separator.setMargin(0)
            separator.setAlignment(Qt.AlignCenter)
            self.scroll_layout.addWidget(separator)

    def show_level_dialog(self, fence, level=None):
        dialog = QDialog(self)
        dialog.setWindowTitle(f"Рівень
{fence.fence_type}.{len(fence.levels) + 1}" if level is None else
f"Редагування рівня {level.material.material_type}")

        form_layout = QFormLayout(dialog)

        materials = get_materials()
        material_combo = QComboBox(dialog)
        material_combo.addItem(materials)

        thickness_input = QLineEdit(dialog)

        if level:
            material_combo.setCurrentText(f"{level.material.material_type}:
{level.material.thermal_conductivity}")
            thickness_input.setText(str(level.thickness))

            add_button = QPushButton("Зберегти", dialog)
            add_button.clicked.connect(lambda: self.save_level(
                fence, level, material_combo.currentText(),
thickness_input.text(), dialog
            ))

            delete_button = QPushButton("Видалити", dialog)

```

```

        if level:
            delete_button.clicked.connect(lambda:
self.delete_level(fence, level, dialog))

        form_layout.addRow("Матеріал:", material_combo)
        form_layout.addRow("Товщина:", thickness_input)
        form_layout.addRow(add_button)
        form_layout.addRow(delete_button)

        dialog.setLayout(form_layout)
        dialog.exec_()

    def delete_level(self, fence, level, dialog):
        fence.levels.remove(level)
        self.save_building_data()
        self.update_scroll_area()
        self.print_building_data()
        dialog.accept()

    def save_level(self, fence, level, material, thickness,
dialog):
        if not thickness:
            self.show_error("Будь ласка, заповніть поле
товщини!")
            return

        if not is_valid_number(thickness):
            self.show_error("Будь ласка, введіть коректне
число у полі товщини.")
            return

        material_name, thermal_conductivity =
material.split(": ")
        new_material = Material(material_name,
float(thermal_conductivity))
        new_level = Level(new_material, float(thickness))

        if level:
            level.material = new_material
            level.thickness = float(thickness)
        else:
            fence.add_level(new_level)

        self.save_building_data()
        self.update_scroll_area()
        self.print_building_data()
        dialog.accept()

    def print_building_data(self):
        print("---- Поточна структура будівлі ----")
        for fence_index, fence in
enumerate(self.building.fences, start=1):
            print(f"Конструкція {fence_index}:
{fence.fence_type}, площа = {fence.area} м2, внутрішня температура

```

```

= {fence.interior_temperature} °C, зовнішня температура =
{fence.exterior_temperature} °C")
        for level_index, level in enumerate(fence.levels,
start=1):
            print(f"    Рівень {fence_index}.{level_index}:
матеріал = {level.material.material_type}, теплопровідність =
{level.material.thermal_conductivity}, товщина = {level.thickness}
м")
            print("-----")

    def show_error(self, message):
        error_dialog = QDialog(self)
        error_dialog.setWindowTitle("Помилка")
        error_label = QLabel(message, error_dialog)
        error_button = QPushButton("OK", error_dialog)
        error_button.clicked.connect(error_dialog.accept)
        layout = QVBoxLayout(error_dialog)
        layout.addWidget(error_label)
        layout.addWidget(error_button)
        error_dialog.setLayout(layout)
        error_dialog.exec_()
    def save_building_data(self):
        try:
            save_data_fence(self.building, self.filename)
        except Exception as e:
            self.show_error(f"Помилка при збереженні даних:
{e}")
    def load_building_data(self):
        try:
            load_data_fence(self.building, self.filename)
            self.update_scroll_area()
            self.print_building_data()
        except Exception as e:
            self.show_error(f"Помилка при завантаженні даних:
{e}")

    def clear_data(self):
        self.building.fences.clear()
        self.save_building_data()
        self.update_scroll_area()
        self.print_building_data()
        self.show_error("Всі дані були очищені!")

```

A.2 – Код методу generate_report

```

def generate_report(self):
    if not hasattr(self, 'results'):
        msg = QMessageBox()
        msg.setIcon(QMessageBox.Warning)
        msg.setText("Розрахунки не виконані.")
        msg.setWindowTitle("Помилка")

```

```

        msg.exec_()
        return

        with open("json_data/building_data_fence.json", "r",
encoding="utf-8") as f:
            building_data = json.load(f)

            fences = building_data.get('fences', [])

            pdf = FPDF()
            pdf.add_page()

            pdf.add_font('DejaVu', '', 'fonts/DejaVuSans.ttf',
uni=True)
            pdf.add_font('DejaVu', 'B', 'fonts/DejaVuSans-
Bold.ttf', uni=True)
            pdf.add_font('DejaVu', 'I', 'fonts/DejaVuSans-
Oblique.ttf', uni=True)
            pdf.set_font('DejaVu', '', 14)

            pdf.set_font("DejaVu", 'B', 16)
            pdf.cell(180, 10, txt="Детальний звіт розрахунків",
ln=True, align='C')

            pdf.set_font("DejaVu", 'I', 14)
            pdf.cell(180, 10, txt=f"Рік будівлі:
{self.building_year}", ln=True)
            pdf.cell(180, 10, txt=f"Загальна теплова потужність:
{self.results['total_load']:.2f} Вт", ln=True)
            pdf.cell(180, 10, txt=f"Потужність від конструкцій:
{self.results['construction_load']:.2f} Вт", ln=True)
            pdf.cell(180, 10, txt=f"Потужність від вентиляції:
{self.results['ventilation_load']:.2f} Вт", ln=True)

            pdf.ln(5)

            pdf.set_font("DejaVu", 'B', 14)
            pdf.cell(180, 10, txt="Деталі конструкцій:", ln=True,
align='C')
            pdf.set_font("DejaVu", '', 14)

            for i, fence in enumerate(fences, start=1):
                pdf.multi_cell(180, 10, txt=f"{i}. Тип
конструкції: {fence['fence_type']}", ln=True)
                pdf.cell(180, 10, txt=f"    Площа: {fence['area']}
м²", ln=True)
                pdf.cell(180, 10, txt=f"    Внутрішня температура:
{fence['interior_temperature']}°C", ln=True)
                pdf.cell(180, 10, txt=f"    Зовнішня температура:
{fence['exterior_temperature']}°C", ln=True)

                for j, level in enumerate(fence['levels'],
start=1):

```

```

        pdf.cell(180, 10, txt=f"    {i}.{j} Матеріал:
{level['material']}, Товщина: {level['thickness']} м", ln=True)

        pdf.ln(5)

        pdf.line(10, pdf.get_y(), 200, pdf.get_y())

        file_dialog = QFileDialog()
        file_dialog.setDefaultSuffix(".pdf")
        file_path, _ = file_dialog.getSaveFileName(None,
"Зберегти звіт", "", "PDF файли (*.pdf)")

        if file_path:
            pdf.output(file_path)
            print(f"Звіт збережено як {file_path}.")
        else:
            print("Не вибрано файл для збереження.")

```