ЗВІТ З ПЕРЕВІРКИ НА ПЛАГІАТ

ЦЕЙ ЗВІТ ЗАСВІДЧУЄ, ЩО ПРИКРПЛЕНА РОБОТА

Сердечна О КІ-228

БУЛА ПЕРЕВІРЕНА СЕРВІСОМ ДЛЯ ЗАПОБІГАННЯ ПЛАГІАТУ

MY.PLAG.COM.UA І МАЄ:

СХОЖІСТЬ

1%

РИЗИК ПЛАГІАТУ

12%

ПЕРЕФРАЗУВАННЯ

0%

НЕПРАВИЛЬНІ ЦИТУВАННЯ

0%

Назва файлу: Сердечна О КІ-228.docx

Файл перевірено: 2023-05-31

Звіт створено: 2023-05-31

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПрАТ (77.93.36.128) «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ» (essuir.sumdu.edu.ua)

Кафедра (www.zieit.edu.ua) Інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Зав.кафедрою _____
д.е.н.,доцент Левицький С.І.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

РОЗРОБКА СИСТЕМИ РОЗРАХУНКУ ТЕПЛОВОЛОГІСТНОГО РЕЖИМУ ОГОРОДЖУВАЛЬНОЇ КОНСТРУКЦІЇ

Виконав
ст. гр. КІ-228 _____ О.М. Сердечна
(підпис)

Керівник
к.т.н. _____ О.А. Хараджян
(підпис)

Запоріжжя

2023

ПРАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ

ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра (77.93.36.128) Інформаційних технологій (essuir.sumdu.edu.ua)

ЗАТВЕРДЖУЮ
Зав. кафедрою (docplayer.net)

_____
д.е.н.,доцент Левицький С.І.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

Студенту гр. *КІ – 228*, спеціальності *«Комп'ютерна інженерія»*

Сердечна Оксана Миколаївна

1.Тема: *Розробка системи розрахунку тепловологістного режиму огороджувальної конструкції.*

затверджена наказом по інституту № 02-10 27.01 2023 р.

2. Термін здачі студентом закінченої роботи: 12.06. 2023 р.

3. Перелік питань, що (docplayer.net) підлягають розробці:

1. Аналіз впливу тепловологістного стану огороджувальних конструкцій на приміщення.
2. Аналіз шляхів обміну вологи у огороджувальних конструкціях.
3. Аналіз норм для розрахунку тепловологістного стану.
4. Розробка алгоритму розрахунку тепловологістного стану за ДСТУ.

5. Розробка структур даних та програмних класів.
6. Розробка користувацького інтерфейсу програми.
7. Розробка програмних модулів користувальницького інтерфейсу.
8. Розробка інтерфейсу діалогових вікон.
9. Виконання тестових розрахунків.

Дата видачі завдання: 16.01.2023 р.

Студент		О.М. Сердечна
	(підпис)	(прізвище та ініціали)
Керівник роботи		О.А. Хараджян
	(підпис)	(прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна (eprints.library.odeku.edu.ua) бакалаврська робота містить 50 сторінок, 3 рисунки, 3 додатки, 7 використаних джерел.

Об’єкт роботи: тепловологістний стан огороджувальних конструкціїй.

Предмет роботи: програми для розрахунку тепловологістного стану огороджувальної конструкціїї.

Мета роботи: розробка програми для розрахунку тепловологістного стану огороджувальної конструкціїї з урахуванням її шарів.

Задачі роботи: аналіз впливу тепловологістного стану огороджувальних конструкцій на приміщення; аналіз шляхів обміну вологи у огороджувальних конструкціях; аналіз норм для розрахунку тепловологістного стану; алгоритм розрахунку тепловологістного стану за ДСТУ; розробка структур даних та програмних класів; розробка користувацького інтерфейсу програми; розробка програмних модулів користувальницького інтерфейсу; розробка інтерфейсу діалогових вікон; виконання тестових розрахунків.

У багатьох країнах будівлі споживають більше енергії, ніж транспорт і промисловість. За статистикою Міжнародного енергетичного агентства, у всьому світі будівництво відповідає за більше споживання електроенергії, ніж будь-який інший сектор.

Розроблений програмний засіб дозволяє виконати розрахунок складних огороджувальних конструкці будівлі, що проектується або експлуатується, на накопичення вологи у елементах конструкції та можливості її конденсації

ОГОРОДЖУВАЛЬНА КОНСТРУКЦІЯ, ПАРЦІАЛЬНИЙ ТИСК, ТЕПЛОВОЛОГІСТНИЙ СТАН, QT, QXML

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ (MYDISSER.COM)

Скорочення	Повна назва	Пояснення/переклад
МЕА	Міжнародне енергетичне агентство	

ВСТУП

Більше 90% нашого часу ми проводимо в будівлях, тобто в офісі або вдома. Енергія, яку споживають в будівлях (житлових і комерційних), складає значну частку загального споживання енергії в країні. Ця частка сильно залежить від ступеня електрифікації, рівня урбанізації, площі забудови на душу населення, переважаючого клімату, а також національної і місцевої політики для підвищення енергоефективності.

У багатьох країнах будівлі споживають більше енергії, ніж транспорт і промисловість. За статистикою Міжнародного енергетичного агентства, у всьому світі будівництво відповідає за більше споживання електроенергії, ніж будь-який інший сектор.

Будівельна галузь має одну характерну якість – вона має високий ступень регулювання. Будівельні норми часто впливають на використання матеріалів і технологій. Стандарти, як обов’язкові, так і добровільні, мають значний вплив на енергоефективність. Таким чином, регулятивні режими, якщо вони існують, можуть забезпечити шлях до підвищення енергетичної ефективності будівель.

Об’єкт роботи: тепловологістний стан огороджувальних конструкціїй.

Предмет роботи: програми для розрахунку тепловологістного стану огороджувальної конструкціїї.

Мета роботи: розробка програми для розрахунку тепловологістного стану огороджувальної конструкціїї з урахуванням її шарів.

Задачі роботи:

- аналіз впливу тепловологістного стану огороджувальних конструкцій на приміщення;
- аналіз шляхів обміну вологи у огороджувальних конструкціях;
- аналіз норм для розрахунку тепловологістного стану;
- алгоритм розрахунку тепловологістного стану за ДСТУ;
- розробка структур даних та програмних класів;
- розробка користувацького інтерфейсу програми;

- розробка програмних модулів користувальницького інтерфейсу;
- розробка інтерфейсу діалогових вікон;
- виконання тестових розрахунків.

РОЗДІЛ 1
ТЕРМОВОЛОГІСТНИЙ СТАН ОГОРОДЖУВАЛЬНИХ КОНСТРУКЦІЙ

1.1. Тепловологістний стан огороджувальних конструкцій та його вплив на приміщення

Більше 90% нашого часу ми проводимо в будівлях, тобто в офісі або вдома. Енергія, яку споживають в будівлях (житлових і комерційних), складає значну частку загального споживання енергії в країні. Ця частка сильно залежить від ступеня електрифікації, рівня урбанізації, площі забудови на душу населення, переважаючого клімату, а також національної і місцевої політики для підвищення енергоефективності.

У багатьох країнах будівлі споживають більше енергії, ніж транспорт і промисловість. За статистикою Міжнародного енергетичного агентства (МЕА), у всьому світі будівництво відповідає за більше споживання електроенергії, ніж будь-який інший сектор.

Будівельна галузь має одну характерну якість – вона має високий ступень регулювання. Будівельні норми часто впливають на використання матеріалів і технологій. Стандарти, як обов’язкові, так і добровільні, мають значний вплив на енергоефективність. Таким чином, регулятивні режими, якщо вони існують, можуть забезпечити шлях до підвищення енергетичної ефективності будівель.

Одна з проблем будівель, яку вирішують нормативними , організаційними та технічними шляхами, це стан вологості огороджувальних конструкцій.

Проблема в тому, що багато будівель мають старі огороджувальних конструкцій. Волога всередині огороджувальних конструкцій з часом викликає прогресуюче псування матеріалів, плями та цвіль на стінах, що негативно впливає на мікрокліматичні умови навколишнього середовища та здоров'я.

Значна присутність вологи в стінах значно знижує ступінь теплової ізоляції стіни до такої міри, що, розсіювання тепла в навколишнє середовище може зрости до 65%, що спричиняє дискомфорт і збільшення витрат на опалення.

Вологість у цегляній кладці може проявлятися різними способами, здебільшого через різні причини. Основні шляхи надходження вологи до огороджувальних конструкцій: структурна вода, баланс вологості, будівельна вологість, аварійна вологість, метеорологічна волога, конденсаційна вологість, капілярна волога.

Структурна вода складається з частини води, яка хімічно зв'язана з будівельними матеріалами. Таким чином, структурна вода тісно пов'язана з матеріалом і не має відношення до деградації.

Баланс вологості - це вміст вологи в матеріалах, що знаходяться в термодинамічній рівновазі з навколишнім середовищем. Поверхня матеріалів поглинає вологу у вигляді водяної пари безпосередньо з середовища, в якому вони знаходяться, і в залежності від пористості матеріалів. Чим більше частина дрібних пор, тим більше буде гігроскопічність матеріалу. Очевидно, що в середовищі з високою відносною вологістю рівноважна вологість матеріалу буде більшою. Крім того, температура навколишнього середовища та наявність гігроскопічних солей впливатимуть на параметри, що може вплинути на поведінку матеріалу, значно підвищуючи вологість.

Будівельна вологість спричинена водою, яка використовується під час приготування розчину, штукатурки та бетону. Надлишок води має тенденцію до випаровування під час фази твердіння, доводячи вологість конструкції до природних значень. Цей тип вологи виникає лише на етапі будівництва будівлі або в кінці реконструкції та реставрації, поступово зникаючи з часом.

Випадкова вологість зазвичай виникає через поломку або несправність систем водопостачання та опалення, конструкцій покрівлі, зовнішніх дверей. Цю вологість зазвичай легко визначити, вона з'являється в тих частинах

будівлі, які безпосередньо контактують з місцем витоку, і її, як правило, легко та швидко усунути.

Атмосферна вологість є прямим впливом, пов’язаним з атмосферними опадами. Вона частіше зустрічається на зовнішніх поверхнях будівель, а також може з’являтися на обмежених ділянках, таких як виступи або підвіконня. Здебільшого це залежить від напрямку вітру та пористості будівельних матеріалів. Фасадна штукатурка відіграє першочергову роль, оскільки її основна функція – обмежити проникнення дощової води, яка потрапляє на внутрішню поверхню виключно через тріщини, що проходять крізь стіни. Тому атмосферна вологість є поверхневою і зменшується протягом кількох годин після випадання опадів у результаті випаровування.

Конденсаційна вологість виникає внаслідок збільшення поверхневої та проміжної щільності кладки, викликаного переходом пари, присутньої в навколишньому середовищі, з газоподібного стану в рідкий. Конденсат з’являється у вигляді дрібних водяних крапель на водонепроникних покриттях і у вигляді темних плям на пористих матеріалах, особливо в місцях з інтенсивним утворенням пари. Це явище відбувається там, де температура поверхні нижча за точку роси.

Інтерстиціальна конденсація пов’язана з дифузією водяної пари, що проникає крізь стіни, які розділяють температуру навколишнього середовища та різну вологість, і виникає у внутрішніх шарах стіни, якщо вони зроблені неправильно. Таким чином, конденсаційна вологість пов’язана з кількістю водяної пари, присутньої в навколишньому середовищі, поганою теплоізоляцією, погодними умовами в той час, недостатньою паропроникністю та технічними помилками в послідовності шарів стін. Будучи настільки безпосередньо пов’язаним із термічною та гігрометричною динамікою навколишнього середовища, як зовнішньої, так і внутрішньої, вона має тривати протягом тривалого часу, перш ніж спричинити значне збільшення вологи в кладці настільки, щоб з’явилися ознакм збільшення ступінь вологості.

Підйомна волога або капілярна дія відбувається, коли вода з землі піднімається вгору крізь цеглу та будівельний розчин за допомогою капілярної дії.

Явище має більш-менш очевидні чинники, що залежать від багатьох факторів, таких як кількість води, присутньої в землі, розмір капілярів стінових матеріалів, наявність гідроізоляції в кладці та здатність до випаровування зовнішніх і внутрішніх поверхонь. кладки.

Водорозчинні солі, як правило, пов'язані з висхідною вологою, а також сульфати, нітрати та хлориди, які, досягнувши поверхні, кристалізуються внаслідок випаровування води. Капілярну вологу, що піднімається, легко діагностувати, спостерігаючи за наявністю явних водяних плям на стінах від підлоги вгору, які часто супроводжуються наявністю білястих висолів через відкладення на поверхні водорозчинних солей.

Таким чином, вологість стін є основною причиною погіршення якості з подальшим зниженням комфорту проживання.

1.2. Норми для розрахунку тепловологістного стану

Рішення для підвищення теплотехнічних характеристик огороджувальних конструкцій, застосування систем опалення з регульованими тепловими режимами при грамотному підході забезпечить високий рівень безпеки та комфорту для життя людини.

В енергозбереженні велике значення надається мінімізації втрат тепла через огороджувальні конструкції.

Утеплення зовнішніх стін ефективними теплоізоляційними матеріалами забезпечить необхідний рівень термостійкості та відновлення нормальних параметрів мікроклімату в житлові приміщення.

В даний час для забезпечення нормативних параметрів використовуються різні будівельні матеріали і технології енергоефективності житлових будинків.

Перенесення вологи є дуже складним процесом і знання механізмів перенесення вологи, властивості матеріалу, початкові умови та граничні умови часто обмежені.

Критична вологість поверхні, яка може призвести до таких проблем, як утворення цвілі на внутрішніх поверхнях будівель.

Інтерстиціальна конденсація всередині огороджувальної конструкції в опалювальні періоди, коли внутрішня температура зазвичай вища, ніж зовнішня, а також в періоди охолодження, коли внутрішня температура зазвичай нижча, ніж зовнішня.

Оцінка часу, необхідного для висихання шарів огороджувальної конструкції між шарами з високою паронепроникністю, після зволоження з будь-якого джерела та ризик утворення міжшарової конденсації в інших місцях конструкції під час процесу сушіння.

У деяких випадках потік повітря з внутрішньої частини будівлі в огороджувальну конструкцію є основним механізмом для транспортування вологи, що може значно збільшити ризик виникнення проблем із конденсатом.

Метод визначення температури внутрішньої поверхні будівельного компонента або будівельного елемента, нижче якої ймовірне зростання цвіль, враховуючи внутрішню температуру та відносну вологість можна використовувати щоб оцінити ризик інших проблем із конденсацією внутрішньої поверхні.

Оцінка ризику міжшарової конденсації внаслідок дифузії водяної пари не враховує низку важливих фізичних явищ, зокрема:

- зміна властивостей матеріалу з вмістом вологи;
- капілярне всмоктування та перенесення рідкої вологи всередині матеріалів;
- рух повітря зсередини будівлі до компонента через щілини або всередині повітряних прошарків;
- гігроскопічна вологоємність матеріалів.

РОЗДІЛ 2
РОЗРОБКА АЛГОРИТМУ РОЗРАХУНКУ ТЕПЛОВОЛОГІСТНОГО СТАНУ

2.1. Алгоритм розрахунку тепловологістного стану

Проектування огороджувальних конструкцій будівлі виконується так, щоб запобігти несприятливому впливу критичних вологість поверхні.

Розглянемо методи та алгоритми розрахунку, які будуть використані у програмі розрахунку.

Конденсація на поверхні може спричинити пошкодження незахищених будівельних матеріалів, чутливих до вологи. Його можна приймати тимчасово і в невеликих кількостях, наприклад на вікнах і плитці у ванних кімнатах, якщо поверхня не вбирає вологу, і вживаються відповідні заходи для запобігання її контакту з прилеглою поверхнею чутливих жо вологи матеріалів. Існує ризик розвитку цвілі, якщо середньомісячна відносна вологість поверхні перевищує критичну відносна вологість $\varphi_{si,cr}$, яку слід приймати як 0.8.

Окрім зовнішніх параметрів (температура повітря та вологість повітря), на конденсацію вологи впивають наступні чинники:

- «теплова якість» кожного елемента огороджувальної конструкції, представлена термічним опором, термічним мости, геометрія та опір внутрішньої поверхні. Термічну якість можна охарактеризувати температурний коефіцієнт на внутрішній поверхні f_{Rsi};
- внутрішнє вологопостачання;
- внутрішня температура повітря та система опалення та її налаштування.

Щоб уникнути розвитку цвілі, середньомісячна відносна вологість на поверхні не повинна перевищувати критичну відносну вологість вологість φ_{sicr},

яку слід прийняти як 0.8. Якщо необхідно уникнути корозії використовується інший критерій $\varphi_{sicr} \leq 0.6$.

Основні етапи процедури проектування полягають у визначенні внутрішньої вологості повітря, а потім, на основі необхідної відносної вологості на поверхні, розраховується об'єм вологи ν_{sat}, або тиск насичення пари p_{sat}. З цього значення отримують мінімальну температуру поверхні і необхідну «теплова якість» огороджувальних конструкцій.

Для кожного місяця року виконується наступне:

- визначення зовнішньої температури;
- визначення зовнішньої вологісті;
- визначення внутрішньої температури;
- визначення внутрішньої відносної вологості;
- при максимально прийнятній відносній вологості на поверхні, $\varphi_{si} = \varphi_{sicr}$, розраховується мінімальний допустимий тиск насиченої пари, p_{sit}

$$p_{sat}(\theta_{si}) = \frac{p_i}{\phi_{sicr}} \tag{2.1}$$

- визначення мінімально прийнятну температуру поверхні, $\theta_{si,min}$, від мінімально прийнятної тиск насиченої пари, обчислений в е);
- від мінімально прийнятної температури поверхні $\theta_{si,min}$, передбачуваної внутрішньої температури повітря θ_i і зовнішньої температури θ_e, обчислюється мінімальний температурний фактор $f_{Rsi,min}$.

Місяць із найвищим необхідним значенням $f_{Rsi,min}$ є критичним. Температурний фактор для цього місяця є $f_{Rsi,max}$, і будівельний елемент має бути спроектований таким чином, щоб $f_{Rsi,max}$ завжди перевищувався

$f_{Rsi} > f_{Rsi,макс}$.

Оцінка поверхневої конденсації на елементах з низькою тепловою інерцією, таких як, наприклад, вікна і їхні рами, які швидко реагують на зміни температури, потребують іншої процедури.

Конденсат на внутрішній поверхні віконних рам може стати незручністю, якщо вода стікає на прилеглі елементи конструкції і може спричинити корозію металевих каркасів або гниття дерев'яних конструкцій. Максимально допустима відносна вологість на поверхні рами $\varphi_{si} = 1$.

Деяка періодична конденсація на віконних рамах може бути прийнятною, незалежно від зазначеної процедури нижче обмежить це можливо за наступним алгоритмом.

- Визначення зовнішньої температури як середнє за кілька років найнижче середньодобове значення температури в кожному році.
- Визначення внутрішньої температури;
- Визначення внутрішньої відносної вологісті.
- При максимально прийнятній відносній вологості на внутрішній поверхні $\varphi_{si} = 1{,}0$, розрахувати мінімально прийнятний тиск пари, p_{sit}

$$p_{sat}(\theta_{si}) = p_i \tag{2.2}$$

- Визначити мінімально прийнятну температуру поверхні $\theta_{si,min}$, від мінімально прийнятного тиску насиченої пари.
- Від мінімально прийнятної температури поверхні $\theta_{si,min}$, передбачуваної внутрішньої температури повітря θ_i і зовнішньої температури θ_e, обчислюється необхідний температурний коефіцієнт будівельного елемента $f_{Rsi,min}$.

Завдяки складній формі та різноманітності матеріалів, які використовуються у віконних рамах, і взаємодії між ними скло, рама та стіна, що містить вікно, теплові потоки та температура поверхні, як правило, не можуть розраховуватися простими одновимірними методами.

Починаючи з першого місяця, в якому передбачається будь-яка конденсація, місячне середнє зовнішнє значення умови використовуються для розрахунку кількості конденсації або випаровування в кожному з 12 місяців року. Накопичена маса конденсованої води в кінці тих місяців, коли є конденсація відбулася порівнюється із загальним випаровуванням протягом решти року. Припускаються одновимірні стаціонарні умови. Єдиним розглянутим ефектом руху повітря є наявність повітряної порожнини, яка добре вентилюється назовні. Вплив руху повітря через будівельну систему не враховується.

Перенесення вологи вважається чистою дифузією водяної пари, що описується наступним рівнянням:

$$g = \frac{\delta_0}{\mu}\frac{\Delta p}{d} = \delta_0 \frac{\Delta p}{s_d} \tag{2.3}$$

де $\delta_0 = 2 \times 10{-}10$ кг/(м·с·Па).

Щільність теплового потоку визначається як

$$q = \lambda \frac{\Delta \theta}{d} = \frac{\Delta \theta}{R} \tag{2.4}$$

де λ - теплопровідність;

R - тепловий опір.

При розрахунках необхідно враховувати джерела помилок, які виникають у результаті спрощень. Теплопровідність залежить від вмісту вологи, при конденсація/випаровуванні тепло виділяється/поглинається, що змінить розподіл температури та значення температури насичення і впливає на кількість волиги при конденсації/висиханні. Капілярне всмоктування та перенесення рідини вологи відбуваються в багатьох матеріалах, і це може змінити розподіл

вологи. Рух повітря всередині будівельних матеріалів, щілин, швів або повітряних просторів може змінити розподіл вологісті за допомогою конвекції. Справжні граничні умови не є постійними протягом місяця.

Будівельний елемент розділяють на ряд однорідних шарів з паралельними сторонами та визначаються властивості матеріалу кожного шару та поверхневі коефіцієнти. Кожен шар в багатошаровій конструкції або компоненти, які включають будь-які шари з облицюванням або покриттями, повинні бути обчислюватись як окремий шар, повністю враховуючи їх відповідні властивості теплопередачі та пропускання вологи. Обчислюється термічний опір R і повітряний шар, еквівалентний дифузії водяної пари товщина s_d кожного окремого шару будівельного елемента. Рекомендується, щоб елементи з термічним опором більше 0,25 м2·К/Вт поділялися на ряд умовних шарів, кожен з термічним опором не більше 0,25 м2·К/Вт.

Деякі матеріали, такі як листовий метал, ефективно перешкоджають проходженню водяної пари, а отже мають нескінченне значення μ. Однак для розрахунку необхідне кінцеве значення μ для матеріалу. Для цих матеріалів слід прийняти μ = 100 000.

Розрахунок термічного опору та повітряного шару, що еквівалентний дифузії водяної пари.

$$R'_n = R_{se} + \sum_{j=1}^{n} R_j \tag{2.5}$$

$$s'_{d,n} = \sum_{j=1}^{n} s_{d,j} \tag{2.6}$$

Загальний термічний опір і товщина повітряного шару, що еквівалентна дифузії водяної пари, визначаються за формулами (2.7) і (2.8):

$$R'_T = R_{si} + \sum_{j=1}^{N} R_j + R_{se} \tag{2.7}$$

$$s'_{d,T} = \sum_{j=1}^{N} s_{d,j} \tag{2.8}$$

Температура на кожній поверхні розділу між матеріалами обчислюється за формулою:

$$\theta'_n = \theta_e + \frac{R'_n}{R'_T}(\theta_i - \theta_e) \tag{2.9}$$

Розподіл температури в кожному шарі є лінійним, враховуючи припущення про стаціонарні умови,

Для наглядного представлення на графіку будується поперечний переріз елемента будівлі з товщиною кожного шару, еквівалентною товщина повітряного шару дифузії водяної пари s_d. Прямими лініями з'єднують точки тиска насичення пари на кожній поверхні розділу між матеріалами.

Якщо немає накопиченого конденсату за попередній місяць, будується профіль тиску пари як пряма лінія між внутрішнім і зовнішнім тиском пари (p_i і p_e). Якщо ця лінія не перевищує тиск насичення на будь-якій межі розділу, конденсація не відбувається.

Швидкість потоку пари через будівельний елемент можна розрахувати як:

$$g = \delta_0 \frac{p_i - p_e}{s'_{d,T}} \tag{2.10}$$

Якщо тиск пари перевищує тиск насичення на будь-якій межі розділу, припустимо, що локальне значення тиску пари дорівнює тиску насичення, і перебудовують лінію тиску пари як ряд лінії, які дотикаються, але не

перевищують профіль тиску насиченої пари в якомога меншій кількості точокю. Ці точки є межами розділу конденсації.

Швидкість конденсації - це різниця між кількістю вологи, що транспортується до та кількість вологи, що транспортується з поверхні розділу конденсації:

$$g_c = \delta_0 \left(\frac{p_i - p_c}{s'_{d,T} - s'_{d,c}} - \frac{p_c - p_e}{s'_{d,c}} \right) \tag{2.11}$$

У будівельному компоненті з більш ніж однією площиною конденсації виконується облік кількості конденсації на кожній поверхні розділу.

Швидкість конденсації розраховується для кожної межі конденсації з різниці нахилу між послідовними прямими.

На поверхні c1:

$$g_{c1} = \delta_0 \left(\frac{p_{c2} - p_{c1}}{s'_{d,c2} - s'_{d,c1}} - \frac{p_{c1} - p_e}{s'_{d,c1}} \right) \tag{2.12}$$

На поверхні c2:

$$g_{c2} = \delta_0 \left(\frac{p_i - p_{c2}}{s'_{d,T} - s'_{d,c2}} - \frac{p_{c2} - p_{c1}}{s'_{d,c2} - s'_{d,c1}} \right) \tag{2.13}$$

Коли є конденсат, накопичений за попередні місяці, на одній або кількох поверхнях розділу, тиск пари має дорівнювати тиску насичення, а профіль тиску пари має бути накреслений як прямі лінії між значеннями, що представляють внутрішній тиск пари, межі розділу конденсації та зовнішній тиск пари. Якщо значення тиску пари перевищують значення насичення в будь-якій площині, перемалюйте лінії тиску пари.

Швидкість випаровування розраховується як

$$g_{ev} = \delta_0 \left(\frac{p_i - p_c}{s'_{d,T} - s'_{d,c}} - \frac{p_c - p_e}{s'_{d,c}} \right) \tag{2.14}$$

За домовленістю, конденсація відбувається, якщо вираз додатний, а випаровування, якщо вираз від'ємний.

У будівельній конструкції з більш ніж однією межею конденсації розраховується швидкість випаровування для кожної площини окремо.

Швидкості випаровування для двох поверхонь розділу випаровування розраховуються наступним чином.

На поверхні c1:

$$g_{ev1} = \delta_0 \left(\frac{p_{c2} - p_{c1}}{s'_{d,c2} - s'_{d,c1}} - \frac{p_{c1} - p_e}{s'_{d,c1}} \right) \tag{2.15}$$

На поверхні c2:

$$g_{ev2} = \delta_0 \left(\frac{p_i - p_{c2}}{s'_{d,T} - s'_{d,c2}} - \frac{p_{c2} - p_{c1}}{s'_{d,c2} - s'_{d,c1}} \right) \tag{2.16}$$

Якщо накопичена кількість конденсату на межі розділу наприкінці місяця розраховується як від'ємне значення, або встановіть його на нуль, або обчисліть час, протягом якого накопичений конденсат досягне нуля, а потім розділіть місяць на дві частини з конденсатом на межі розділу та без нього.

У будівельномій конструкції з більш ніж одною площиною конденсації можуть бути місяці з конденсацією на одній поверхні розділу та випаровування на іншій.

Критерії, що використовуються для оцінки будівельних компонентів

У результаті розрахунків робиться один з трьох висновків:

- конденсація не передбачається на будь-якій межі в будь-який місяць. Структура вільна від інтерстиціальної конденсації;

- конденсація відбувається на одній або кількох межах розділу протягом кількох місяців, але для кожної відповідної поверхні розділу немає чистого накопичення протягом року, оскільки прогнозується, що весь конденсат знову випарується. У цьому випадку розраховується максимальна кількість конденсату, що виникла на кожній межі розділу, і місяць, протягом якого відбувся максимум. Крім того, ризик стікання води або деградації будівлі матеріалів і погіршення теплових характеристик як наслідок розрахункового максимуму кількість вологи слід враховувати відповідно до нормативних вимог та інших вказівок у стандарти продукції. матеріалів буде дуже високим.
- конденсація на одній або кількох поверхнях розділу не повністю випаровується. У цьому випадку структура не проходить розрахунок, додатково розраховується максимальна сума вологи, що виникла на кожній межі розділу разом із кількістю вологи, що залишилася після дванадцяти місяців у кожній площині конденсації.

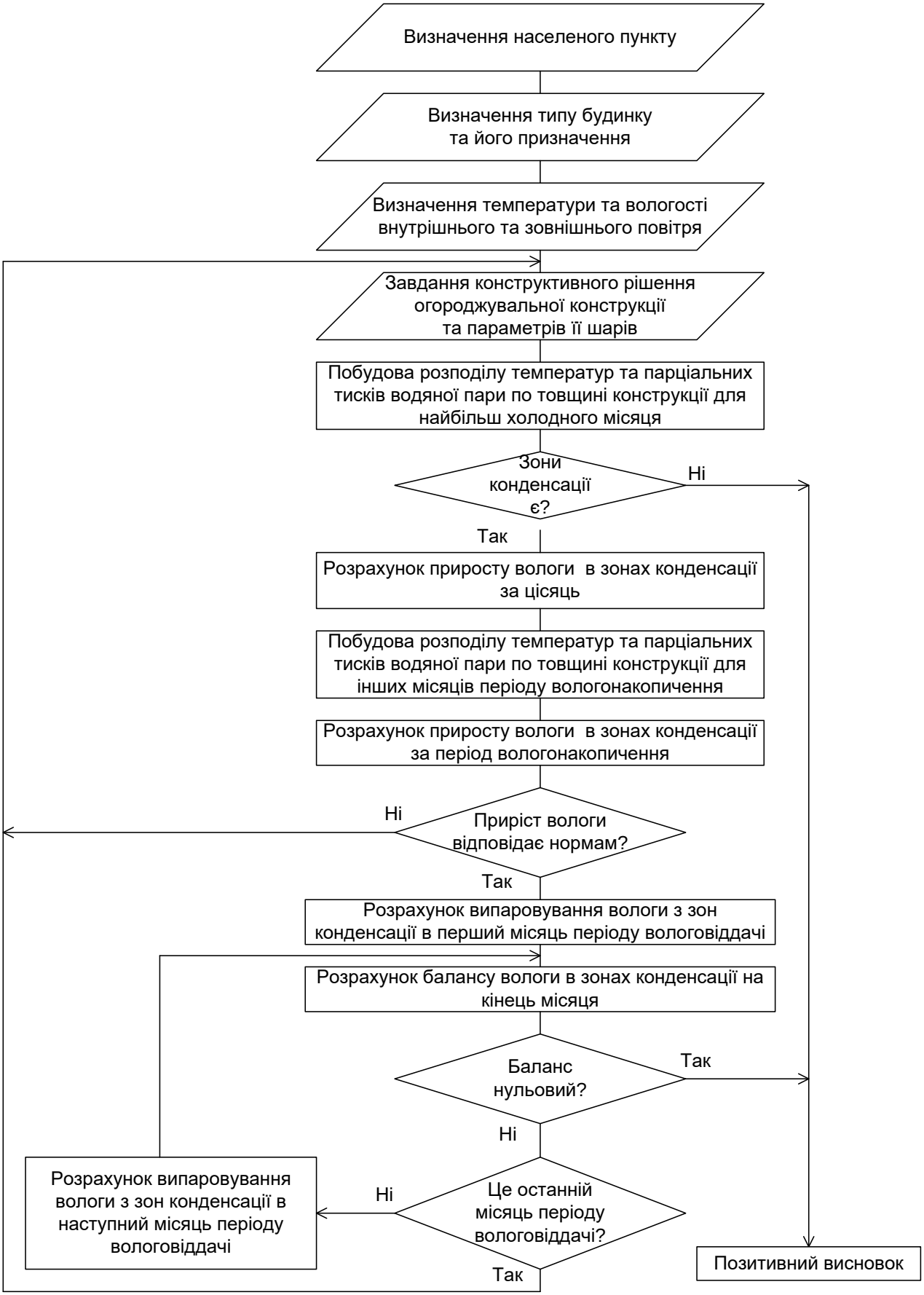

Рис. 2.1 – Загальний алгоритм обчислень

2.2. Структури даних та програмні класи

Структура даних програми складається з наступних класів:

Building - загальна структура будівлі;

Material - будівельні матеріали;

ClimateCondition - кліматичні умови;

Wall - огороджувальні конструкції;

WallSection – однорідні секції огороджувальних конструкцій;

WallLayer – шари секції огороджувальних конструкцій.

Загальна структура будівлі розміщується в класі class Building.

```
class Building
{
public:
    Building();
```

Методи завантаження конфігурації системи

```
    long LoadXML(QString file_name);
    long LoadXML_Materials(QString file_name){return
materials.LoadXML(file_name);}
```

Методи ініціалізації та обчислень

```
    void Init();
    void Calculation();
    void Calculation_Rtermo(double Text, double Hext, double
Tint, double Hint);
    long ClimetCond_GetIndex_by_Id(long Id);
    double get_Temp_InternalLayerWall(long wall_idx);
    double get_Temp_ExternalLayerWall(long wall_idx);
```

Для роботи системи необхідні три об’єкта. База даних будівельних матеріалів

```
    Material materials;
```

Масив властивостей огороджувальних конструкцій

```
    WallVector_t walls;
```

Зовнішні та внутрішні кліматичні умови

```
    ClimateConditionVector_t clime_cond;
};
```

Розглянемо докладно кожен з цих об’єктів.

Клас кліматичних умов відображає стан повітря в заданому об’ємі приміщення

```
class ClimateCondition
{
public:
    ClimateCondition();
    long id; // Ідентифікатор кліматичних умов
    double Temp; // Температура повітря
    double Humidy; // Відносна вологість повітря
    double Wind_Speed; // Швидкість руху повітря
    double Wind_Psi; // Напрям руху повітря
    double Qin; // Потужність джерел тепла у цьому об’ємі повітря
};
```

Для представлення кліматичних умов у різних частинах програми використовується тип масива елементів класу ClimateCondition

```
typedef QVector<ClimateCondition> ClimateConditionVector_t;
```

Властивості будівельного матеріалу представлені структурою Material_obj_t. Для усіх властивостей матеріалів є можливість визначити їх у функції густини, густину – у функції температури.

```
typedef struct
{
    QString name; // Назва матеріалу
    long Id; // Ідентифікатор матеріалу
    QVector<double> Ro; // Густина матерілу
    QVector<double> Q; // Питома теплоємність
    QVector<double> Lambda; // Питома теплопровідність
    QVector<double> R_termo  ; // Опір теплопередачі
    QVector<double> mu; // Коефіцієнт паропроникнення
    QVector<double> R_nu; // Опір паропроникнення
}Material_obj_t;
```

Для збереження та зчитування бази даних матеріалів використовується клас Material, який надає відповідні інтерфейси

```
class Material
{
```

Масив внутрішнього представлення матеріалів

```
    QVector<Material_obj_t> materials_arr;
```

Методи конструювання об’єкту та його ініціалізації

```
public:
    Material();
    ~Material();
    long LoadXML(QString file_name);
```

Методи для доступу до властивостей матеріалу

```
    double Get_Lambda(long index);
    double Get_Lambda_id(long id);
    double Get_R_termo(long index);
    double Get_R_termo_id(long id);
    double Get_mu(long index);
    double Get_mu_id(long id);
    double Get_R_nu(long index);
    double Get_R_nu_id(long id);
};
```

Огороджувальні конструкції представлені класом Wall.

Для доступу до бази будівельних матеріалів в класі є посилання на відповідний зовнішний об’єкт.

```
class Wall
{
    Material *material_ptr;
```

Методи створення та ініціалізації об’єкта

```
public:
    Wall();
    void SetMaterialArr(Material *p);
```

Методи розрахунку теплового потоку крізь огороджувальну конструкцію

```
        double Calculation(double T_air_in, double T_air_out,
double V);
        void Calculation_LayerTerm(double T_air_in, double
T_air_out, double H_air_in, double H_air_out, double V);
```

Параметри огороджувальної конструкції представлені нижче

```
        long Id; // Ідентифікатор огороджувальної конструкції
        QString Name; // Назва огороджувальної конструкції
        long climate_in_id; // Ідентифікатор клімата для
зовнішнього повітря
        long climate_out_id; // Ідентифікатор клімата для
внутрішнього повітря
        WallSectionVector_t WallSections; // Перелік однорідних
секцій огороджувальної конструкції
        double Qin_out; // Тепловий потік кріз конструкцію
    };
```

Для представлення огороджувальних конструкцій у різних частинах програми використовується тип масива елементів класу Wall

```
typedef QVector<Wall> WallVector_t;
```

Властивості однорідних елементів огороджувальної конструкції представлені класом WallSection.

Для доступу до бази будівельних матеріалів в класі є посилання на відповідний зовнішний об’єкт.

```
class WallSection
{
    Material *material_ptr;
public:
```

```
        WallSection();
        void SetMaterialArr(Material *p){material_ptr = p;}
```

Основні властивості секції огороджувальної конструкції представлені наступним кодом

```
        QString Name; // Назва секції
        double H; // Висота секції
        double L; // Ширина секції
        double S; // Площа секції
        WallLayerVector_t WallLayers; // Перелік шарів конструкції
```

Методи розрахунку секції огороджувальної конструкції

```
        void Calculation(long WallType, double T_air_in, double T_air_out, double V, double C12);
        void CalculationLayer(long WallType, double T_air_in, double T_air_out, double H_air_in, double H_air_out, double V, double C12);
```

Внутрішні розрахункові велечини

```
        double R_termo; // Опір теплопередачі
        double Rin, Rout; / Опір на внутрішній та зовнішній поверхнях
        double Tau_in, Tau_out; // Температура на внутрішній та зовнішній поверхнях
        double d_sum; // Товщина елемента конструкції
        double Qin_out; // Тепловий потік крізь конструкцію
    };
```

Для представлення секції огороджувальних конструкцій у різних частинах програми використовується тип масива елементів класу WallSection.

```
typedef QVector<WallSection> WallSectionVector_t;
```

Властивості шарів секції огороджувальної конструкції представлені у класі WallLayer.

```
class WallLayer
{
public:
    WallLayer();
```

Основні властивості шару визначаються наступними даними

```
    long Material_Id; // Ідентифікатор будівельно матеріалу
    double delta; // Товщина шару
    double Termo; // Температура на поверхні шару
    double Psat_max; // Тиск насичення водяної пари на поверхні
    double Psat_val; // Тиск водяної пари на поверхні
};
```

Для представлення шарів огороджувальних конструкцій у різних частинах програми використовується тип масива елементів класу WallLayer.

```
typedef QVector<WallLayer> WallLayerVector_t;
```

Наведена структура даних забезпечує розрахунок тепловологістного стану елементів огороджувальних конструкцій.

РОЗДІЛ 3
РОЗРОБКА КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ ПРОГРАМИ

3.1. Структура програмних модулів користувальницького інтерфейсу

Програма розроблена з використанням фреймворка Qt.

Розглянемо основні методи розроблених класів.

Метод розрахунку теплового опрору та температур в однорідних секціях огороджувальних конструкцій. Метод приймає на вході тип огороджувальної конструкції WallType, температуру повітря в приміщенні T_air_in, зовнішню температуру повітря T_air_out, швидкість руху повітря V.

```
    void WallSection::Calculation(long WallType, double T_air_in,
double T_air_out, double V, double C12)
    {
        long i;
        double lambda;
        R_termo = 0.0;
        if(!material_ptr) return;
```

Розрахунок виконується для кожного шару огороджувальної конструкції

```
        for(i=0; i<WallLayers.size()-1; i++)
        {
            lambda          =          material_ptr->Get_Lambda_id(
WallLayers[i].Material_Id );
            if( lambda>1e-6 )
                R_termo += WallLayers[i].delta / lambda;
            else
```

```
                R_termo      +=      material_ptr->Get_R_termo_id(
WallLayers[i].Material_Id );
        }
```

Якщо тип огороджувальної конструкції відповідає підлозі по грунту, то виконується спеціальний розрахунок по зонам підлоги

```
        if( WallType == WALL_FLOOR )
        {
            S = FloorCalc_S(La, Lb, Lc, Ld);
            R_termo = FloorCalc_Rtermo(La, Lb, Lc, Ld, R_termo);
        }
```

Розрахунок температури на площинах шару

```
        if(R_termo>1e-6)
        {
            Tau_in      =      T_air_in      -      (T_air_in      -
T_air_out)*Rin/(R_termo+Rin+Rout);
            Tau_out      =      T_air_in      -      (T_air_in      -
T_air_out)*(Rin+R_termo)/(R_termo+Rin+Rout);
        }
        else
        {
            Tau_in = T_air_in;
            Tau_out = T_air_out;
        }
```

Розрахунок теплового опору для внутрішньої та зовнішньої поверхонь огороджувальної конструкції

```
        Rin  =  1.0/Calc_Alpha(WallType,  WALL_SUFACE_IN,  T_air_in,
Tau_in, C12, 0.0);
```

```
        Rout    =    1.0/Calc_Alpha(WallType,    WALL_SUFACE_OUT,
T_air_out, Tau_out, C12, V);
    }
```

Метод розрахунку параметрів шару будівельної конструкції. Метод приймає на вході тип огороджувальної конструкції WallType, температуру повітря в приміщенні T_air_in, зовнішню температуру повітря T_air_out, відносну вологість повітря внутрішню H_air_in, відносну вологість повітря зовнішню H_air_out, швидкість руху повітря V.

```
    void  WallSection::CalculationLayer(long  WallType,  double
T_air_in, double T_air_out, double H_air_in, double H_air_out,
double V, double C12)
    {
        long i;
        double lambda;
        double R_termo_local = 0.0;
        if(!material_ptr) return;
        double E_int, E_ext;
        double R_nu_sum = 0.0, R_nu = 0.0;
        double mu;
```

Розрахунок парціального тиску водяної пари для внутрішнього та зовнішнього повітря

```
        E_int = H_air_in * 0.01 * PsatT(T_air_in);
        E_ext = H_air_out * 0.01 * PsatT(T_air_out);
```

Розрахунок вологопроникності шарів конструкції

```
        for(i=0; i<WallLayers.size()-1; i++)
        {
```

```
            mu                    =                    material_ptr-
>Get_mu_id(WallLayers[i].Material_Id);
            if( mu>1e-6 )
                R_nu += WallLayers[i].delta / mu;
            else
                R_nu               +=               material_ptr-
>Get_R_nu_id(WallLayers[i].Material_Id);
        }
```

Розрахунок температури у площинах шарів та парціального тиску водяної пари при заданих умовах та парціального тиску насичення водяної пари.

```
        for(i=0; i<WallLayers.size(); i++)
        {
            if(WallLayers[i].Material_Id==0)
            { // розрахунок для зовнішньої поверхні
                WallLayers[i].Termo  =  T_air_in  -  (T_air_in  -
T_air_out)*(Rin+R_termo_local)/(R_termo+Rin+Rout);
                R_termo_local += WallLayers[i].delta / lambda;
                WallLayers[i].Psat_max                          =
PsatT(WallLayers[i].Termo);
                WallLayers[i].Psat_val  =  E_int  -  (E_int  -
E_ext)*R_nu_sum/R_nu;
            }
            else
            {// розрахунок для внутрішніх поверхонь
                lambda       =       material_ptr->Get_Lambda_id(
WallLayers[i].Material_Id );
                if( lambda>1e-6 )
                {
                    WallLayers[i].Termo = T_air_in - (T_air_in -
T_air_out)*(Rin+R_termo_local)/(R_termo+Rin+Rout);
                    R_termo_local += WallLayers[i].delta / lambda;
                }
                else
```

```
                {
                    WallLayers[i].Termo = T_air_in - (T_air_in -
T_air_out)*(Rin+R_termo_local)/(R_termo+Rin+Rout);
                    R_termo_local += material_ptr->Get_R_termo_id(
WallLayers[i].Material_Id );
                }
```

Розрахунок парціального тиску водяної пари та парціального тиску насичення водяної пари.

```
                WallLayers[i].Psat_max                          =
PsatT(WallLayers[i].Termo);
                WallLayers[i].Psat_val   =   E_int   -   (E_int   -
E_ext)*R_nu_sum/R_nu;
                mu                =                material_ptr-
>Get_mu_id(WallLayers[i].Material_Id);
                if( mu>1e-6 )
                    R_nu_sum += WallLayers[i].delta / mu;
                else
                    R_nu_sum          +=          material_ptr-
>Get_R_nu_id(WallLayers[i].Material_Id);
            }
```

Розрахунок умови виникнення конденсату

```
            if( WallLayers[i].Psat_val > WallLayers[i].Psat_max)
                WallLayers[i].SatAlarm = 1;
            else
                WallLayers[i].SatAlarm = 0;
        }
    }
```

Розрахунок теплового опору огороджувальної конструкції. Метод приймає на вході температуру повітря в приміщенні T_air_in, зовнішню температуру повітря T_air_out, швидкість руху повітря V.

```
    double Wall::Calculation(double T_air_in, double T_air_out, double V)
    {
        double C12 = 4.5;
        long i;
        Qin_out=0.0;
        for( i=0; i<WallSections.size(); i++)
        {
            WallSections[i].Calculation( WallType, T_air_in, T_air_out, V, C12);
            WallSections[i].Qin_out = WallSections[i].S*(T_air_in - T_air_out)/(WallSections[i].R_termo + WallSections[i].Rin + WallSections[i].Rout);
            Qin_out -= WallSections[i].Qin_out;
        }
        return Qin_out;
    }
```

Розрахунок параметрів огороджувальної конструкції. Метод приймає на вході температуру повітря в приміщенні T_air_in, зовнішню температуру повітря T_air_out, відносну вологість повітря внутрішню H_air_in, відносну вологість повітря зовнішню H_air_out, швидкість руху повітря V.

```
    void Wall::Calculation_LayerTerm(double T_air_in, double T_air_out, double H_air_in, double H_air_out, double V)
    {
        double C12 = 4.5;
        long i;
        for( i=0; i<WallSections.size(); i++)
        {
```

```
            WallSections[i].CalculationLayer( WallType, T_air_in,
T_air_out, H_air_in, H_air_out, V, C12);
        }
    }
```

Ініціалізація розрахунку огороджувальних конструкцій усієї будівлі виконується наступним методом

```
    void Building::Init()
    {
        long i;
        for(i=0; i<walls.size(); i++)
        {
            walls[i].SetMaterialArr(&materials);
            walls[i].climate_in_idx                                =
ClimetCond_GetIndex_by_Id(walls[i].climate_in_id);
            walls[i].climate_out_idx                               =
ClimetCond_GetIndex_by_Id(walls[i].climate_out_id);
        }
    }
```

Розрахунок теплових потоків та потоків вологи для усіх приміщень будівлі виконується ітераційно наступним чином

```
    void Building::Calculation()
    {
        long i, j;
        double Q, sum;
        do
        {
            for(i=0; i<walls.size(); i++)
            {

walls[i].Calculation(clime_cond[walls[i].climate_in_idx].Temp,
```

```
clime_cond[walls[i].climate_out_idx].Temp,
clime_cond[walls[i].climate_out_idx].Wind_Speed);
            }
            sum=0.0;
            for(i=0; i<clime_cond.size(); i++)
            {
                if(clime_cond[i].Type                                    ==
ClimateCondition::TYPE_IN)
                {
                    Q=0.0;
                    for(j=0; j<walls.size(); j++)
                    {
                         if( walls[j].climate_in_idx == i)
                        {
                            Q += walls[j].Qin_out;
                        }
                    }
                    Q += clime_cond[i].Qin;
                    sum += Q ;
                    clime_cond[i].Temp += Q*0.01;
                }
            }
        }
        while(fabs(sum)>1.0);
```

Після розрахунку теплових потоків виконується кінцевий розрахунок температури площин шарів огороджуючих конструкцій

```
        for(i=0; i<walls.size(); i++)
        {
            walls[i].Calculation_LayerTerm(clime_cond[
walls[i].climate_in_idx].Temp,
clime_cond[walls[i].climate_out_idx].Temp,
```

```
                    clime_cond[walls[i].climate_in_idx].Humidy,
clime_cond[      walls[i].climate_out_idx].Humidy,      clime_cond[
walls[i]. climate_out_idx ].Wind_Speed);
        }
    }
```

Обчислені значення зберігаються у локальних масивах даних. Відображення виконується в таблицях для кожного елементу конструкції.

3.2. Інтерфейс діалогових вікон

Інтерфейс програми реалізовано за допомогою віджетів Qt. Ініціалізація інтерфейсу та об’єктів для розрахунку виконується в конструкторі MainWindow

```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
```

Створення об’єктів графічного інтерфейсу

```
    ui->setupUi(this);
```

Завантаження бази даних будівельних матеріалів та структури конструкцій

```
    building.LoadXML("..\\TermoCalc\\building.xml");
    building.LoadXML_Materials("..\\TermoCalc\\material.xml");
    building.Init();
```

Розрахунок тепловологістного режиму

```
        building.Calculation();
```

Відображення результатів розрахунку виконується у таблиці. Спочатку ініціалізуються основні параметри таблиці.

```
        QStringList str_list;
        str_list<<"T, 0C"<<"Qin";
        ui->tableWidget_Climet-
>setRowCount(building.clime_cond.size());
        ui->tableWidget_Climet->setColumnCount(2);
        ui->tableWidget_Climet-
>setHorizontalHeaderLabels(str_list);
        ui->tableWidget_Climet->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);
```

А потім виконується заповнення комірок таблиці

```
        long i, j;
        QString str;
        for(i = 0; i<building.clime_cond.size(); i++)
        {
            str.setNum(building.clime_cond[i].Temp);
            ui->tableWidget_Climet->setItem(i,       0,       new
QTableWidgetItem(str));
            str.setNum(building.clime_cond[i].Qin);
            ui->tableWidget_Climet->setItem(i,       1,       new
QTableWidgetItem(str));
        }
```

Для відображення структури огороджувальних конструкцій та їх параметрів створюється дерево QTreeWidgetItem.

```
        QTreeWidgetItem *tr_item1;
        QTreeWidgetItem *tr_item2;
        QString str_1;
        wall_section_pos_t wall_section_pos;
        for(i=0; i<building.walls.size(); i++)
        {
            tr_item1 = new QTreeWidgetItem;
            str=QString("wall%1                                    -
%2").arg(i).arg(building.walls[i].Name);
            tr_item1->setText(0, str);
            wall_section_pos.wall_pos = i;
            wall_section_pos.section_pos = -1;
            QVariant data=QVariant::fromValue( wall_section_pos );
            tr_item1->setData(0, Qt::UserRole, data);
            for(j=0; j<building.walls[i].WallSections.size(); j++)
            {
                tr_item2 = new QTreeWidgetItem;
                str=QString("section%1                                 -
%2").arg(j).arg(building.walls[i].WallSections[j].Name);
                tr_item2->setText(0, str);
                wall_section_pos.wall_pos = i;
                wall_section_pos.section_pos = j;
                 data=QVariant::fromValue( wall_section_pos );
                tr_item2->setData(0, Qt::UserRole, data);
                tr_item1->addChild(tr_item2);
            }
            ui->treeWidget_Walls->addTopLevelItem(tr_item1);
        }
```

Для забезпечення інтерактивного відображення параметрів кожної огороджувальної конструкції виконується об’єднання сигналів та сокетів

```
        connect(                              ui->treeWidget_Walls,
SIGNAL(itemPressed(QTreeWidgetItem*,int)),                    this,
SLOT(TreeWall_itemPressed(QTreeWidgetItem*,int)) );
```

```
}
```

Обробник повідомлення itemPressed для таблиці дозволяє побудувати графік залежності парціального тиску водяної пари.

```
void MainWindow::TreeWall_itemPressed(QTreeWidgetItem *item, int column)
{
```

Визначається індекс огороджувальної конструкції та індекс секції в локальних таблицях

```
        QVariant variant = item->data(column, Qt::UserRole);
        long wall_idx = variant.value<wall_section_pos_t>().wall_pos;
        long section_idx = variant.value<wall_section_pos_t>().section_pos;
        if(wall_idx<0 || section_idx<0)
            return;
```

Визначається загальна товщина конструкції

```
       double d_sum = building.walls[wall_idx].WallSections[section_idx].d_sum;
```

Виконується ініціалізація графіка

```
    QVector<double> x_len, y_Temperature, y_H2Opres_max, y_H2Opres_val;
        ui->widgetPlot->clearPlottables();
        ui->widgetPlot->clearGraphs();
```

Створення точок на внутрішніх шарах

```
        for(i=0;     i<building.walls[wall_idx     ].WallSections[
section_idx ].WallLayers. size(); i++)
        {
            x_len.append(dd);
            d     =     building.walls[     wall_idx].WallSections[
section_idx]. WallLayers[ i].delta;
            dd+=d;
            d     =     building.walls[wall_idx].     WallSections[
section_idx].WallLayers[ i].Termo;
            y_Temperature.append(d);
            d  =  building.walls[wall_idx].WallSections[section_idx
].WallLayers[ i].Psat_max;
            y_H2Opres_max.append(d);
            d = building.walls[wall_idx].WallSections[ section_idx
]. WallLayers[ i].Psat_val;
            y_H2Opres_val.append(d);
        }
```

Створення останньої точки на зовнішній поверхні

```
       QString str_text;
        str_text      =      QString("delta_sum=%1      m\nQin_out
section=%2\nR_termo=%3").arg(dd)
    .arg(building.walls[wall_idx].WallSections[section_idx].Qin_ou
t)
    .arg(building.walls[wall_idx].WallSections[section_idx].R_term
o);
        ui->label_2->setText(str_text);
```

Створення роздільних ліній

```
        QCPCurve *verticalLine;
        QVector<double> xx(2) , yy(2);
```

```
        for(i=0; i<y_Temperature.size(); i++)
        {
            verticalLine = new QCPCurve(ui->widgetPlot->xAxis, ui-
>widgetPlot->yAxis);
            xx[0]        =        x_len[i];        yy[0]        =
building.get_Temp_InternalLayerWall(wall_idx);
            xx[1]        =        x_len[i];        yy[1]        =
building.get_Temp_ExternalLayerWall(wall_idx);
            verticalLine->setData(xx, yy);
            verticalLine->brush().setColor(Qt::black);
        }
```

Додавання до графіку даних температур та тиску та його оновлення

```
        ui->widgetPlot->addGraph();
        ui->widgetPlot->graph(0)->setData(x_len, y_Temperature);
        ui->widgetPlot->addGraph();
        ui->widgetPlot->graph(1)->setData(x_len, y_H2Opres_max);
        ui->widgetPlot->graph(1)->brush().setColor(Qt::red);
        ui->widgetPlot->addGraph();
        ui->widgetPlot->graph(2)->setData(x_len, y_H2Opres_val);
        ui->widgetPlot->graph(2)->brush().setColor(Qt::green);
        ui->widgetPlot->rescaleAxes();
        ui->widgetPlot->replot();
```

Ініціалізація таблиці з даними про шари матеріалу

```
        QString str;
        ui->tableWidget->clear();
        QStringList str_list;
        str_list<<"delta, m"<<"Termo, oC"<<"Psat max, Pa"<<"Psat
val, Pa";
        ui->tableWidget->setColumnCount(4);
        ui->tableWidget->setHorizontalHeaderLabels(str_list);
```

```
        ui->tableWidget->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);
        ui->tableWidget-
>setRowCount(building.walls[wall_idx].WallSections[section_idx].Wa
llLayers.size());
```

Заповнення комірок таблиці

```
        dd=0.0;
        for(i   =   0;   i<building.walls[wall_idx].WallSections[
section_idx]. WallLayers. size(); i++)
        {
            str.setNum(dd);
    dd  +=  building.walls[wall_idx].WallSections[  section_idx  ].
WallLayers[ i].delta;
    ui->tableWidget->setItem(i, 0, new QTableWidgetItem( str));
    str.setNum(building.walls[wall_idx].WallSections[section_idx].
WallLayers[ i].Termo);
    ui->tableWidget->setItem(i, 1, new QTableWidgetItem( str));
    str.setNum(building.walls[wall_idx].WallSections[section_idx].
WallLayers[i].Psat_max);
    ui->tableWidget->setItem(i, 2, new QTableWidgetItem( str));
    str.setNum(building.walls[wall_idx].WallSections[section_idx].
WallLayers[i].Psat_val);
    ui->tableWidget->setItem(i, 3, new QTableWidgetItem(str));
        }
    }
```

Функція обробник для кнопки розрахунку виконує розрахунок теплового опрору кожного шару та його відображення у таблиці. Розрахунок ведеться для заданих користувачем умов.

```
    void MainWindow::on_pushButton_Calc_Rtermo_clicked()
    {
```

```
        int i, j;
        building.Init();
        double Text, Hext, Tint, Hint;
        Text = ui->lineEdit_Text->text().toDouble();
        Hext = ui->lineEdit_Hext->text().toDouble();
        Tint = ui->lineEdit_Tint->text().toDouble();
        Hint = ui->lineEdit_Hint->text().toDouble();
```

Розрахунок теплового опрору

```
        building.Calculation_Rtermo(Text, Hext, Tint, Hint);
```

Ініціалізація таблиці для відображення даних

```
        QString str;
        str = QString("%1 Вт").arg(building.Qin_out_sum);
        ui->label_Qin_out_sum->setText(str);
        ui->tableWidget_Rtermo->clear();
        QStringList str_list;
        str_list<<"N"<<"Стена/Секция"<<"Толщина,     м"<<"R_termo,
0C*W/m2"<<"P, Вт";
        ui->tableWidget_Rtermo->setColumnCount(5);
        ui->tableWidget_Rtermo-
>setHorizontalHeaderLabels(str_list);
        ui->tableWidget_Rtermo->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);
        long cnt=0;
        for(i=0; i<building.walls.size(); i++)
            cnt+=building.walls[i].WallSections.size();
    ui->tableWidget_Rtermo->setRowCount(  building.walls.size()  +
cnt );
```

Відображення даних по кожній огороджувальній конструкції

```
        int row=0;
        QTableWidgetItem *item;
        QFont font;
        for(i = 0; i<building.walls.size(); i++)
        {
            str = QString("%1:").arg(i);
            item = new QTableWidgetItem(str);
            font = item->font();
            font.setPointSize(10);
            font.setBold(true);
            item->setFont(font);
            ui->tableWidget_Rtermo->setItem(row, 0, item);
            item = new QTableWidgetItem(building.walls[i].Name);
            font = item->font();
            font.setPointSize(10);
            font.setBold(true);
            item->setFont(font);
            ui->tableWidget_Rtermo->setItem(row, 1, item);
            str.setNum( building.walls[i].Qin_out );
            item = new QTableWidgetItem(str);
            font = item->font();
            font.setPointSize(8);
            font.setBold(true);
            item->setFont(font);
            ui->tableWidget_Rtermo->setItem(row, 4, item);
```

Відображення даних по кожній секції огороджувальної конструкції

```
            row++;
    for(j = 0; j<building.walls[i].WallSections.size(); j++)
            {
                str = QString("%1:%2").arg(i).arg(j);
    ui->tableWidget_Rtermo->setItem(row, 0, new QTableWidgetItem(
str));
```

```
        ui->tableWidget_Rtermo->setItem(row, 1, new QTableWidgetItem(
building. walls[i].WallSections[j].Name));
        str.setNum( building.walls[i]. WallSections[j]. R_termo );
        ui->tableWidget_Rtermo->setItem(row, 3, new QTableWidgetItem(
str));
        str.setNum( building.walls[i]. WallSections[j]. Qin_out );
        ui->tableWidget_Rtermo->setItem(row, 4, new QTableWidgetItem(
str));
                        row++;
                    }
                }
            }
```

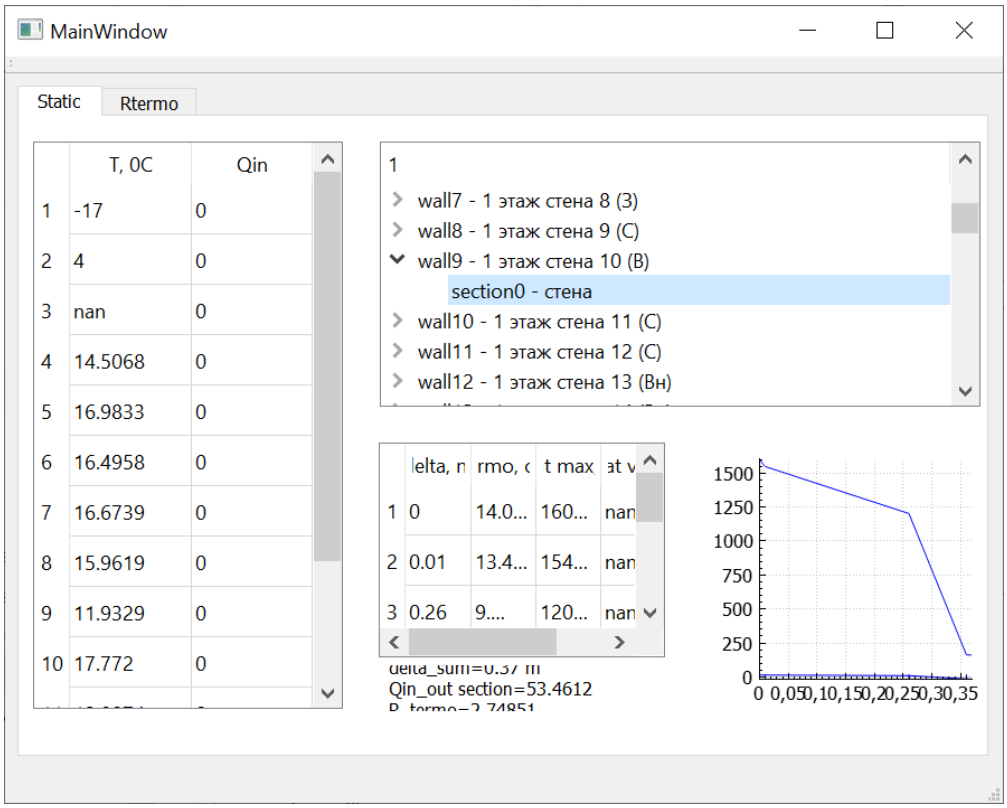

Рис.3.1 - Вікно відображення структури огороджувальної конструкції

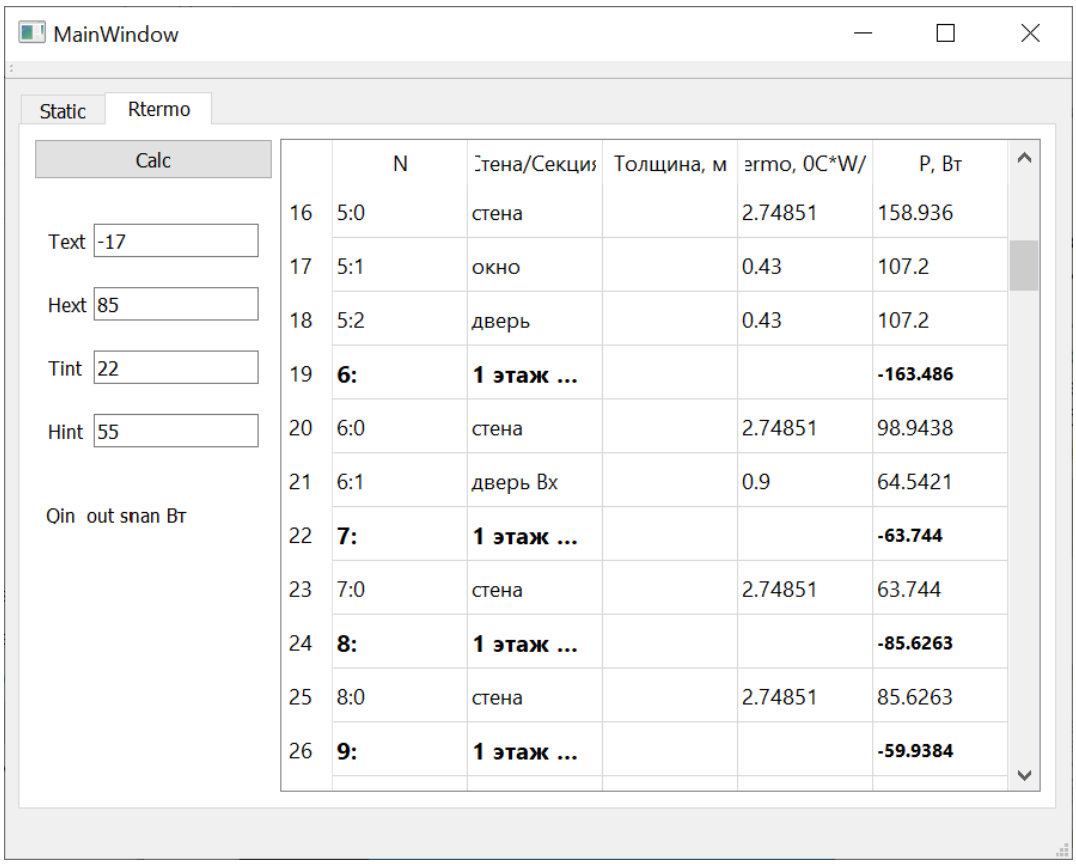

Рис. 3.2 – Вікно відображення теплового опору та теплового потоку для довільних умов

ВИСНОВКИ

Значну частину часу ми проводимо в будівлях, тобто в офісі або вдома. Енергія, яку споживають в будівлі, складає значну частку загального споживання енергії в країні. Ця частка сильно залежить від ступеня електрифікації, рівня урбанізації, площі забудови на душу населення, переважаючого клімату, а також національної і місцевої політики для підвищення енергоефективності. У багатьох країнах будівлі споживають більше енергії, ніж транспорт і промисловість.

Одна з проблем будівель, яку вирішують нормативними , організаційними та технічними шляхами, це стан вологості огороджувальних конструкцій.

Проблема в тому, що багато будівель мають старі огороджувальних конструкцій. Волога всередині огороджувальних конструкцій з часом викликає прогресуюче псування матеріалів, плями та цвіль на стінах, що негативно впливає на мікрокліматичні умови навколишнього середовища та здоров'я.

Значна присутність вологи в стінах значно знижує ступінь теплової ізоляції стіни до такої міри, що, розсіювання тепла в навколишнє середовище може зрости до 65%, що спричиняє дискомфорт і збільшення витрат на опалення.

Основні шляхи надходження вологи до огороджувальних конструкцій: структурна вода, баланс вологості, будівельна вологість, аварійна вологість, метеорологічна волога, конденсаційна вологість, капілярна волога.

В даний час для забезпечення нормативних параметрів використовуються різні будівельні матеріали і технології енергоефективності житлових будинків.

Розроблений програмний засіб дозволяє виконати розрахунок складних огороджувальних конструкці будівлі, що проектується або експлуатується, на накопичення вологи у елементах конструкції та можливості її конденсації.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ-Н Б В.2.6-192:2013 «Настанова з розрахункової оцінки тепловологістного стану огороджувальних конструкцій» [Чинний від 2014-01-01]. Київ, 2014. 41 с.
2. ДБН В.2.6-31:2016 «Теплова ізоляція будівель» (ela.kpi.ua) [Чинний від 2016-10-01]. Київ, 2016. 37 с.
3. Qt Group [Електронний ресурс]. – Режим доступу: www. URL: https://www.qt.io / (дата звернення:14.02.23)
4. QCustomPlot Documentation [Електронний ресурс]. – Режим доступу: www. URL: https://www.qcustomplot.com/index.php/support/documentation (дата звернення:15.02.23)

ДОДАТКИ

ДОДАТОК А

Вихідний код програми mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

#include <QString>
#include <QDomDocument>
#include <QFile>
#include <QFont>

#include "qcustomplot.h"

typedef struct
{
    long wall_pos;
    long section_pos;
} wall_section_pos_t;
Q_DECLARE_METATYPE(wall_section_pos_t);

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

//    building.LoadXML("..\\TermoCalc\\building.xml");
//
building.LoadXML_Materials("..\\TermoCalc\\material.xml");
    building.LoadXML("building.xml");
    building.LoadXML_Materials("material.xml");
    building.Init();
    building.Calculation();

//================================
```

```
        QStringList str_list;
        str_list<<"T, 0C"<<"Qin";
        ui->tableWidget_Climet-
>setRowCount(building.clime_cond.size());
        ui->tableWidget_Climet->setColumnCount(2);
        ui->tableWidget_Climet-
>setHorizontalHeaderLabels(str_list);
        ui->tableWidget_Climet->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);

        long i, j;
        QString str;
        for(i = 0; i<building.clime_cond.size(); i++)
        {
            str.setNum(building.clime_cond[i].Temp);
            ui->tableWidget_Climet->setItem(i,       0,       new
QTableWidgetItem(str));
            str.setNum(building.clime_cond[i].Qin);
            ui->tableWidget_Climet->setItem(i,       1,       new
QTableWidgetItem(str));
        }
        //================================

        QTreeWidgetItem *tr_item1;
        QTreeWidgetItem *tr_item2;
        QString str_1;

        wall_section_pos_t wall_section_pos;

        for(i=0; i<building.walls.size(); i++)
        {
            tr_item1 = new QTreeWidgetItem;
            str=QString("wall%1                                    -
%2").arg(i).arg(building.walls[i].Name);
            tr_item1->setText(0, str);
            wall_section_pos.wall_pos = i;
```

```
            wall_section_pos.section_pos = -1;

            //QVariant data(wall_section_pos);
            QVariant data=QVariant::fromValue( wall_section_pos );
            //QVariant data;
            //data.setValue(wall_section_pos);

            tr_item1->setData(0, Qt::UserRole, data);

            for(j=0; j<building.walls[i].WallSections.size(); j++)
            {
                tr_item2 = new QTreeWidgetItem;
                str=QString("section%1                                -
%2").arg(j).arg(building.walls[i].WallSections[j].Name);
                tr_item2->setText(0, str);
                wall_section_pos.wall_pos = i;
                wall_section_pos.section_pos = j;

    //            QVariant data(wall_section_pos);
                 data=QVariant::fromValue( wall_section_pos );
    //            QVariant data;
    //            data.setValue(wall_section_pos);
                tr_item2->setData(0, Qt::UserRole, data);

                tr_item1->addChild(tr_item2);
            }

            ui->treeWidget_Walls->addTopLevelItem(tr_item1);
        }

        connect(                              ui->treeWidget_Walls,
SIGNAL(itemPressed(QTreeWidgetItem*,int)),                    this,
SLOT(TreeWall_itemPressed(QTreeWidgetItem*,int)) );


    }
```

```
     void  MainWindow::TreeWall_itemPressed(QTreeWidgetItem  *item,
int column)
     {
         QVariant variant = item->data(column, Qt::UserRole);
         long                        wall_idx                        =
variant.value<wall_section_pos_t>().wall_pos;
         long                      section_idx                       =
variant.value<wall_section_pos_t>().section_pos;

         if(wall_idx<0 || section_idx<0)
             return;

         //================================

         double                          d_sum                         =
building.walls[wall_idx].WallSections[section_idx].d_sum;        //
толщина стены

         QVector<double>   x_len,   y_Temperature,   y_H2Opres_max,
y_H2Opres_val;

         ui->widgetPlot->clearPlottables();
         ui->widgetPlot->clearGraphs();



         long i;
         double dd=0.0;
         double d;

         //-------------
         // создание первой точки на внутренней поверхности
         //x_len.append(dd);
```

```
//y_Temperature.append(building.get_Temp_InternalLayerWall(wall_id
x));
          //y_H2Opres_max.append(0.0);
          //y_H2Opres_val.append(0.0);

          // создание точек на внутренних слоях
          for(i=0;
i<building.walls[wall_idx].WallSections[section_idx].WallLayers.si
ze(); i++)
          {
              x_len.append(dd);
              d                                                    =
building.walls[wall_idx].WallSections[section_idx].WallLayers[i].d
elta;
              dd+=d;

              d                                                    =
building.walls[wall_idx].WallSections[section_idx].WallLayers[i].T
ermo;
              y_Temperature.append(d);
              d                                                    =
building.walls[wall_idx].WallSections[section_idx].WallLayers[i].P
sat_max;
              y_H2Opres_max.append(d);
              d                                                    =
building.walls[wall_idx].WallSections[section_idx].WallLayers[i].P
sat_val;
              y_H2Opres_val.append(d);

          }

          // создание последней точки на наружной поверхности
          //x_len.append(dd);
```

```
//y_Temperature.append(building.get_Temp_ExternalLayerWall(wall_id
x));
         //y_H2Opres_max.append(0.0);
         //y_H2Opres_val.append(0.0);

         //-------------
        QString str_text;
         str_text      =      QString("delta_sum=%1      m\nQin_out
section=%2\nR_termo=%3").arg(dd)

.arg(building.walls[wall_idx].WallSections[section_idx].Qin_out)

.arg(building.walls[wall_idx].WallSections[section_idx].R_termo);
         ui->label_2->setText(str_text);



         //--------------------
         // отрисовка разделительных линий
         QCPCurve  *verticalLine;         //  Объявляем  объект  для
вертикальной линии
         QVector<double> xx(2) , yy(2);
         for(i=0; i<y_Temperature.size(); i++)
         {
             verticalLine = new QCPCurve(ui->widgetPlot->xAxis, ui-
>widgetPlot->yAxis);
             xx[0]        =        x_len[i];        yy[0]        =
building.get_Temp_InternalLayerWall(wall_idx);
             xx[1]        =        x_len[i];        yy[1]        =
building.get_Temp_ExternalLayerWall(wall_idx);
             verticalLine->setData(xx,  yy);                  //  И
устанавливаем координаты
             verticalLine->brush().setColor(Qt::black);
         }

         //--------------------
```

```
        ui->widgetPlot->addGraph();
        ui->widgetPlot->graph(0)->setData(x_len, y_Temperature);
        ui->widgetPlot->addGraph();
        ui->widgetPlot->graph(1)->setData(x_len, y_H2Opres_max);
        ui->widgetPlot->graph(1)->brush().setColor(Qt::red);
        ui->widgetPlot->addGraph();
        ui->widgetPlot->graph(2)->setData(x_len, y_H2Opres_val);
        ui->widgetPlot->graph(2)->brush().setColor(Qt::green);
        ui->widgetPlot->rescaleAxes();

        ui->widgetPlot->replot();

        //==============================
        QString str;
        ui->tableWidget->clear();

        QStringList str_list;
        str_list<<"delta, m"<<"Termo, oC"<<"Psat max, Pa"<<"Psat
val, Pa";
        ui->tableWidget->setColumnCount(4);
        ui->tableWidget->setHorizontalHeaderLabels(str_list);
        ui->tableWidget->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);
        ui->tableWidget-
>setRowCount(building.walls[wall_idx].WallSections[section_idx].Wa
llLayers.size());
        dd=0.0;
        for(i                          =                          0;
i<building.walls[wall_idx].WallSections[section_idx].WallLayers.si
ze(); i++)
        {

            str.setNum(dd);
            dd                                                    +=
building.walls[wall_idx].WallSections[section_idx].WallLayers[i].d
elta;
```

```
            ui->tableWidget->setItem(i,           0,           new
QTableWidgetItem(str));

str.setNum(building.walls[wall_idx].WallSections[section_idx].Wall
Layers[i].Termo);
            ui->tableWidget->setItem(i,           1,           new
QTableWidgetItem(str));

str.setNum(building.walls[wall_idx].WallSections[section_idx].Wall
Layers[i].Psat_max);
            ui->tableWidget->setItem(i,           2,           new
QTableWidgetItem(str));

str.setNum(building.walls[wall_idx].WallSections[section_idx].Wall
Layers[i].Psat_val);
            ui->tableWidget->setItem(i,           3,           new
QTableWidgetItem(str));
        }
    }

    MainWindow::~MainWindow()
    {
        delete ui;
    }


    void MainWindow::on_pushButton_Calc_Rtermo_clicked()
    {
        int i, j;


        building.Init();
    //    building.Calculation();
        double Text, Hext, Tint, Hint;
        Text = ui->lineEdit_Text->text().toDouble();
        Hext = ui->lineEdit_Hext->text().toDouble();
```

```
        Tint = ui->lineEdit_Tint->text().toDouble();
        Hint = ui->lineEdit_Hint->text().toDouble();

        building.Calculation_Rtermo(Text, Hext, Tint, Hint);

        //======================
        QString str;
        str = QString("%1 Вт").arg(building.Qin_out_sum);
        ui->label_Qin_out_sum->setText(str);

        ui->tableWidget_Rtermo->clear();

        QStringList str_list;
        str_list<<"N"<<"Стена/Секция"<<"Толщина,     м"<<"R_termo,
0C*W/m2"<<"P, Вт";
        ui->tableWidget_Rtermo->setColumnCount(5);
        ui->tableWidget_Rtermo-
>setHorizontalHeaderLabels(str_list);
        ui->tableWidget_Rtermo->horizontalHeader()-
>setSectionResizeMode(QHeaderView::Stretch);

        long cnt=0;
        for(i=0; i<building.walls.size(); i++)
            cnt+=building.walls[i].WallSections.size();
        ui->tableWidget_Rtermo->setRowCount( building.walls.size()
+ cnt );

        int row=0;
        QTableWidgetItem *item;
        QFont font;
        for(i = 0; i<building.walls.size(); i++)
        {
            str = QString("%1:").arg(i);
            item = new QTableWidgetItem(str);
            font = item->font();
            font.setPointSize(10);
```

```
            font.setBold(true);
            item->setFont(font);
            ui->tableWidget_Rtermo->setItem(row, 0, item);

            item = new QTableWidgetItem(building.walls[i].Name);
            font = item->font();
            font.setPointSize(10);
            font.setBold(true);
            item->setFont(font);
            ui->tableWidget_Rtermo->setItem(row, 1, item);

            str.setNum( building.walls[i].Qin_out );
            item = new QTableWidgetItem(str);
            font = item->font();
            font.setPointSize(8);
            font.setBold(true);
            item->setFont(font);
            ui->tableWidget_Rtermo->setItem(row, 4, item);

            row++;
            for(j  =  0;  j<building.walls[i].WallSections.size();
j++)
            {
                str = QString("%1:%2").arg(i).arg(j);
                ui->tableWidget_Rtermo->setItem(row,    0,    new
QTableWidgetItem(str));
                ui->tableWidget_Rtermo->setItem(row,    1,    new
QTableWidgetItem(building.walls[i].WallSections[j].Name));

                str.setNum(
building.walls[i].WallSections[j].R_termo );
                ui->tableWidget_Rtermo->setItem(row,    3,    new
QTableWidgetItem(str));

                str.setNum(
building.walls[i].WallSections[j].Qin_out );
```

```
                ui->tableWidget_Rtermo->setItem(row,     4,     new
QTableWidgetItem(str));

                row++;
            }
        }
    /*
        for(i                          =                          0;
i<building.walls[wall_idx].WallSections[section_idx].WallLayers.si
ze(); i++)
        {

            str.setNum(dd);
            dd                                                    +=
building.walls[wall_idx].WallSections[section_idx].WallLayers[i].d
elta;
            ui->tableWidget->setItem(i,           0,           new
QTableWidgetItem(str));

str.setNum(building.walls[wall_idx].WallSections[section_idx].Wall
Layers[i].Termo);
            ui->tableWidget->setItem(i,           1,           new
QTableWidgetItem(str));

str.setNum(building.walls[wall_idx].WallSections[section_idx].Wall
Layers[i].Psat_max);
            ui->tableWidget->setItem(i,           2,           new
QTableWidgetItem(str));

str.setNum(building.walls[wall_idx].WallSections[section_idx].Wall
Layers[i].Psat_val);
            ui->tableWidget->setItem(i,           3,           new
QTableWidgetItem(str));
        }
    */
```

```
}
```

ДОДАТОК Б

Вихідний код програми building.cpp

```
#include "building.h"
#include <QString>
#include <QDomDocument>
#include <QFile>
#include "wall.h"
#include "walllayer.h"
#include "wallsection.h"
#include "climatecondition.h"
#include "IF97/IF97.h"


Building::Building()
{
}

long Building::LoadXML(QString file_name)
{
    QFile file(file_name);
    if (!file.open(QIODevice::ReadOnly | QIODevice::Text))
    {
        //emit   Log(tr("Невозможно   открыть   XML-конфиг"),
LOG_LEVEL_ERROR);
        return 1;
    }

    QDomDocument doc;
    if (false == doc.setContent(&file))
    {
        return 2;//QString("bad XML-file: setContent");
    }

//    QString str;
```

```
        QDomNode record_node_section;
        QDomNode record_node_layer;
        QDomNode record_node;
        QDomNode record_node_main;
        QDomElement root = doc.documentElement();
    //    QDomNamedNodeMap attrs;

        //while(false == doc.isNull())
        {
            root = doc.documentElement();
            if(root.tagName() == "building" )
            {
                record_node_main = root.firstChild();
                while( false == record_node_main.isNull() )
                {
                    if(record_node_main.nodeName() == "walls" )
                    {
                        record_node                                 =
record_node_main.firstChild();
                        while (false == record_node.isNull())
                        {
                            if(record_node.nodeName() == "wall")
                            {
                                Wall wall;
                                QString str;

                                wall.Id                             =
record_node.attributes().namedItem("id").toAttr().value().toInt();
                                str                                 =
record_node.attributes().namedItem("type").toAttr().value();
                                // тип стена вертикальная VERT-0,
пол FLOOR-1, потолок ROOF-2
                                if(           str.contains("vert",
Qt::CaseInsensitive))
                                    wall.WallType = 0;
```

```
                                    else   if(   str.contains("floor",
Qt::CaseInsensitive))
                                        wall.WallType = 1;
                                    else   if(   str.contains("roof",
Qt::CaseInsensitive))
                                        wall.WallType = 2;
                                    else wall.WallType = str.toInt();

                                    wall.Psi                          =
record_node.attributes().namedItem("psi").toAttr().value().toDoubl
e();
                                    wall.climate_in_id                =
record_node.attributes().namedItem("clime_in").toAttr().value().to
Int();
                                    wall.climate_out_id               =
record_node.attributes().namedItem("clime_out").toAttr().value().t
oInt();
                                    wall.Name                         =
record_node.attributes().namedItem("name").toAttr().value();

                                    record_node_section               =
record_node.firstChild();
                                    while           (false           ==
record_node_section.isNull())
                                    {

if(record_node_section.nodeName() == "section")
                                        {
                                            WallSection wallsection;

                                            wallsection.H             =
record_node_section.attributes().namedItem("H").toAttr().value().t
oDouble();
                                            wallsection.L             =
record_node_section.attributes().namedItem("L").toAttr().value().t
oDouble();
```

```
                                        wallsection.S           =
record_node_section.attributes().namedItem("S").toAttr().value().t
oDouble();
                                        wallsection.Name        =
record_node_section.attributes().namedItem("name").toAttr().value(
);
                                        wallsection.La          =
record_node_section.attributes().namedItem("La").toAttr().value().
toDouble();
                                        wallsection.Lb          =
record_node_section.attributes().namedItem("Lb").toAttr().value().
toDouble();
                                        wallsection.Lc          =
record_node_section.attributes().namedItem("Lc").toAttr().value().
toDouble();
                                        wallsection.Ld          =
record_node_section.attributes().namedItem("Ld").toAttr().value().
toDouble();

                                        if(fabs(wallsection.S)<1e-
6)
                                        {
                                            wallsection.S       =
wallsection.H * wallsection.L;
                                        }
                                        record_node_layer       =
record_node_section.firstChild();
                                        while      (false      ==
record_node_layer.isNull())
                                        {

if(record_node_layer.nodeName() == "layer")
                                            {
                                                WallLayer
walllayer;
```

```
walllayer.Material_Id                                                =
record_node_layer.attributes().namedItem("material_id").toAttr().v
alue().toInt();
                                                    walllayer.delta  =
record_node_layer.attributes().namedItem("delta").toAttr().value()
.toDouble();


wallsection.WallLayers.append(walllayer);
                                                }
                                                record_node_layer    =
record_node_layer.nextSibling();
                                            }
                                            //++++
                                            //              добавление
дополнительного пустого слоя
                                            WallLayer walllayer;
                                            walllayer.Material_Id  =0;
                                            walllayer.delta  = 0.0;

wallsection.WallLayers.append(walllayer);
                                            //----

wall.WallSections.append(wallsection);
                                        }
                                        record_node_section          =
record_node_section.nextSibling();
                                    }

                                    walls.append(wall);
                                }
                                record_node                          =
record_node.nextSibling();
                            }
                        }
```

```
                    if(record_node_main.nodeName() == "clime_room"
)
                    {
                        record_node                                 =
record_node_main.firstChild();
                        while (false == record_node.isNull())
                        {
                            if(record_node.nodeName() == "clime")
                            {
                                ClimateCondition clime;
                                QString str;
                                clime.id                            =
record_node.attributes().namedItem("id").toAttr().value().toInt();

                                str                                 =
record_node.attributes().namedItem("type").toAttr().value();
                                if(            str.contains("int",
Qt::CaseInsensitive))
                                    clime.Type = 1;
                                else    if(    str.contains("ext",
Qt::CaseInsensitive))
                                    clime.Type = 0;
                                else clime.Type = str.toInt();

                                clime.Temp                          =
record_node.attributes().namedItem("T").toAttr().value().toDouble(
);
                                clime.Humidy                        =
record_node.attributes().namedItem("H").toAttr().value().toDouble(
);
                                clime.Wind_Psi                      =
record_node.attributes().namedItem("VP").toAttr().value().toDouble
();
```

```
                                    clime.Wind_Speed                    =
record_node.attributes().namedItem("V").toAttr().value().toDouble(
);
                                    clime.Qin                           =
record_node.attributes().namedItem("Qin").toAttr().value().toDoubl
e();

                                    clime_cond.append(clime);
                                }
                                record_node                             =
record_node.nextSibling();
                            }
                        }
                        record_node_main                                =
record_node_main.nextSibling();
                    }
                }
            }
            return 0;
        }

        void Building::Init()
        {
            long i;

            for(i=0; i<walls.size(); i++)
            {
                walls[i].SetMaterialArr(&materials);

                walls[i].climate_in_idx                                 =
ClimetCond_GetIndex_by_Id(walls[i].climate_in_id);
                walls[i].climate_out_idx                                =
ClimetCond_GetIndex_by_Id(walls[i].climate_out_id);
            }

        }
```

```
long Building::ClimetCond_GetIndex_by_Id(long id)
{
    long i;
    for( i=0; i<clime_cond.size(); i++)
    {
        if(clime_cond[i].id == id)
            return i;
    }
    return 0;
}

double Building::get_Temp_InternalLayerWall(long wall_idx)
{
    if(wall_idx<0 || wall_idx>walls.size())
        return 0.0;
    return clime_cond[walls[wall_idx].climate_in_idx].Temp;
}

double Building::get_Temp_ExternalLayerWall(long wall_idx)
{
    if(wall_idx<0 || wall_idx>walls.size())
        return 0.0;
    return clime_cond[walls[wall_idx].climate_out_idx].Temp;
}

void Building::Calculation()
{
    long i, j;

    //QVector<double> Q_rooms;
    double Q, sum;

    do
    {
        for(i=0; i<walls.size(); i++)
```

```
            {

walls[i].Calculation(clime_cond[walls[i].climate_in_idx].Temp,
clime_cond[walls[i].climate_out_idx].Temp,
clime_cond[walls[i].climate_out_idx].Wind_Speed);
            }

            sum=0.0;
            for(i=0; i<clime_cond.size(); i++)
            {
                if(clime_cond[i].Type                                    ==
ClimateCondition::TYPE_IN)
                {
                    //clime_cond[i].Temp_prev                            =
clime_cond[i].Temp;
                    Q=0.0;
                    for(j=0; j<walls.size(); j++)
                    {
                         if( walls[j].climate_in_idx == i)
                        {
                            Q += walls[j].Qin_out;
                        }
                    }
                    Q += clime_cond[i].Qin;
                    //Q_rooms.append(Q);

                    sum += Q ;
                    clime_cond[i].Temp += Q*0.01;
                }
            }

        }
        while(fabs(sum)>1.0);

        // расчет температур на поверхности слоев ограждений
        for(i=0; i<walls.size(); i++)
```

```
        {

walls[i].Calculation_LayerTerm(clime_cond[walls[i].climate_in_idx]
.Temp, clime_cond[walls[i].climate_out_idx].Temp,
                    clime_cond[walls[i].climate_in_idx].Humidy,
clime_cond[walls[i].climate_out_idx].Humidy,
clime_cond[walls[i].climate_out_idx].Wind_Speed);
        }
    }
    void Building::Calculation_Rtermo(double Text, double Hext,
double Tint, double Hint)
    {
        long i, j;
        Qin_out_sum =0;
        for(i=0; i<walls.size(); i++)
        {
            if(    clime_cond[walls[i].climate_in_idx].Type    ==
ClimateCondition::TYPE_OUT ||
                clime_cond[walls[i].climate_out_idx].Type      ==
ClimateCondition::TYPE_OUT )
                walls[i].Calculation(Tint,                  Text,
clime_cond[walls[i].climate_out_idx].Wind_Speed);
            Qin_out_sum += walls[i].Qin_out;
        }
        // расчет температур на поверхности слоев ограждений
        for(i=0; i<walls.size(); i++)
        {
            walls[i].Calculation_LayerTerm(Tint, Text,
                    Hint,                                   Hext,
clime_cond[walls[i].climate_out_idx].Wind_Speed);
        }
    }
```

ДОДАТОК В

Вихідний код програми material.cpp

```
#include "material.h"

#include <QString>
#include <QDomDocument>
#include <QFile>

Material::Material()
{
//    materials_arr = new QVector<Material_obj_t>;
}
Material::~Material()
{

}

double Material::Get_Lambda(long index)
{
    return materials_arr[index].Lambda[0];
}

double Material::Get_Lambda_id(long id)
{
    long i;
    for( i=0; i<materials_arr.size(); i++)
    {
        if(materials_arr[i].Id == id)
            return materials_arr[i].Lambda[0];
    }
    return 0;
}

double Material::Get_R_termo(long index)
```

```
{
    return materials_arr[index].R_termo[0];
}

double Material::Get_R_termo_id(long id)
{
    long i;
    for( i=0; i<materials_arr.size(); i++)
    {
        if(materials_arr[i].Id == id)
            return materials_arr[i].R_termo[0];
    }
    return 0;
}

double Material::Get_mu(long index)
{
    return materials_arr[index].mu[0];
}

double Material::Get_mu_id(long id)
{
    long i;
    for( i=0; i<materials_arr.size(); i++)
    {
        if(materials_arr[i].Id == id)
            return materials_arr[i].mu[0];
    }
    return 0;
}

double Material::Get_R_nu(long index)
{
    return materials_arr[index].R_nu[0];
}
```

```
double Material::Get_R_nu_id(long id)
{
    long i;
    for( i=0; i<materials_arr.size(); i++)
    {
        if(materials_arr[i].Id == id)
            return materials_arr[i].R_nu[0];
    }
    return 0;
}

long Material::LoadXML(QString file_name)
{
    QFile file(file_name);
    if (!file.open(QIODevice::ReadOnly | QIODevice::Text))
    {
        //emit   Log(tr("Невозможно   открыть   XML-конфиг"),
LOG_LEVEL_ERROR);
        return 1;
    }

    QDomDocument doc;
    if (false == doc.setContent(&file))
    {
        return 2;//QString("bad XML-file: setContent");
    }

    QString str;
    Material_obj_t Material_obj;
    QDomNode record_node;
    QDomElement root = doc.documentElement();
    double d;

    if(root.tagName() == "materials" )
    {
        record_node = root.firstChild();
```

```
            while (false == record_node.isNull())
            {
                if(record_node.nodeName() == "material")
                {
                    Material_obj.Id = 0;
                    //Material_obj.Lambda = 0;
                    Material_obj.name = "";
                    Material_obj.Lambda.clear();
                    Material_obj.Ro.clear();
                    Material_obj.Q.clear();
                    Material_obj.R_termo.clear();
                    Material_obj.mu.clear();
                    Material_obj.R_nu.clear();
                    QDomNamedNodeMap            attrs            =
record_node.attributes();
                    Material_obj.name                            =
record_node.attributes().namedItem("name").nodeValue();
                    Material_obj.Id                              =
record_node.attributes().namedItem("id").toAttr().value().toInt();
                    str = attrs.namedItem("ro").toAttr().value();

Material_obj.Ro.append(str.section(';',0,0).toDouble());
                    d = str.section(';',1,1).toDouble();
                    if(d>0.0)
                        Material_obj.Ro.append(d);
                    str                                          =
attrs.namedItem("lambda").toAttr().value();

Material_obj.Lambda.append(str.section(';',0,0).toDouble());
                    d = str.section(';',1,1).toDouble();
                    if(d>0.0)
                        Material_obj.Lambda.append(d);
                    str                                          =
attrs.namedItem("R_termo").toAttr().value();

Material_obj.R_termo.append(str.section(';',0,0).toDouble());
```

```
                    d = str.section(';',1,1).toDouble();
                    if(d>0.0)
                        Material_obj.R_termo.append(d);
                    str = attrs.namedItem("q").toAttr().value();

Material_obj.Q.append(str.section(';',0,0).toDouble());
                    d = str.section(';',1,1).toDouble();
                    if(d>0.0)
                        Material_obj.Q.append(d);
                    str = attrs.namedItem("mu").toAttr().value();

Material_obj.mu.append(str.section(';',0,0).toDouble());
                    d = str.section(';',1,1).toDouble();
                    if(d>0.0)
                        Material_obj.mu.append(d);
                    str                                          =
attrs.namedItem("R_nu").toAttr().value();

Material_obj.R_nu.append(str.section(';',0,0).toDouble());
                    d = str.section(';',1,1).toDouble();
                    if(d>0.0)
                        Material_obj.R_nu.append(d);
                    materials_arr.append(Material_obj);
                }
                record_node = record_node.nextSibling();
            }
        }
        else
        {
            return 3;
        }
        return 0;
    }
```