Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Розробка програмного забезпечення для автоматизованого проектування силових трансформаторів»
за освітньою програмою: «» Інженерія програмного забезпечення
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1811»

_____ /Микита ВИСОЧЕНКО/
(підпис студента) (ім'я та прізвище)

Керівник: _____ /доц. Юрій ІВЧЕНКО/
(підпис) (посада, ім'я та прізвище)

Нормоконтролер: _____ /доц. Олена КУРОПЯТНИК/
(підпис) (посада, ім'я та прізвище)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Дніпро – 2022 рік

Ministry of Education and Science of Ukraine
Ukrainian State University of Science and Technologies

Faculty «Computer technologies and systems»

Department «Computer information technology»

Explanatory Note
to Bachelor's Thesis

on the topic: «Software development for automated design of power transformers» according to educational curriculum «»
in the Speciality: «121 Software engineering»

Done by the student of the group ПЗ1811: /Mykyta VYSOCHENKO/

Scientific Supervisor: /Yurii IVCHENKO/

Normative controller: /Olena KUROPYATNYK/

Dnipro – 2022

Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет: «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Рівень вищої освіти: бакалавр
Освітня програма: «»Інженерія програмного забезпечення
Спеціальність: «121 Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри КІТ
_____/Вадим ГОРЯЧКІН/
(підпис)
Дата _____

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра
студенту Височенко Микита Ігорович
1. Тема роботи: «Розробка програмного забезпечення для автоматизованого проектування силових трансформаторів» Керівник роботи: Івченко Юрій Миколайович, доцент затверджені наказом № 77 ст від 08.12.2021

2. Строк подання студентом роботи: __.__.202_ р.
3. Вихідні дані до роботи: _____

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати): вступ, збір вимог до програмного забезпечення, огляд літератури, проектування, тестування та налагодження

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі	31.01.2022 – 05.01.2022	
2	Огляд літератури та аналіз аналогів	16.01.2022 – 24.01.2022	
3	Розробка структур вхідних і вихідних даних	25.01.2022 – 02.02.2022	
4	Визначення вимог до програми. Вибір та обґрунтування мови програмування	03.02.2022 – 10.02.2022	
5	Узгодження та затвердження ТЗ	11.02.2022 – 18.02.2022	
6	Розробка та програмування логіки програми	19.02.2022 – 01.03.2022	
7	Розробка і реалізація інтерфейсу користувача	02.03.2022 – 20.03.2022	
8	Відлагодження програми	21.03.2022 – 24.03.2022	
9	Розробка, узгодження та затвердження програмної документації	25.03.2022 – 03.04.2022	
10	Подання кваліфікаційної роботи до кафедри	04.04.2022– 12.06.2022	
11	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	21.06.2022	

Студент _____ Микита ВИСОЧЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ доц. Юрій ІВЧЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається з 4 розділів:

– Вступ – в даному розділі описується сутність розробки, її

актуальність. Складається з 1 сторінки;

– Збір вимог до програмного забезпечення – у цьому розділи описуються аналоги програми та література по даній предметній області, а також проводиться опит зацікавлених сторін для формування найбільш повних вимог до програмного забезпечення. Складається з 3 сторінок;

– Проектування усієї системи – у цьому розділі проведений огляд вхідних і вихідних даних, формалізація задачі, розробка фізичного проекту, проектування інтерфейсу користувача, аналіз проекту, вибір мови програмування. Складається з 21 сторінок;

– Тестування та налагодження – включає в себе вибір стратегії тестування, опис тестів методами «чорного» та «білого» ящика. Також аналіз помилок їх вплив на систему та вирішення проблеми. Складається з 6 сторінок;

– Висновки. Складається з 1 сторінки;

– Список літератури

– Включає в себе бібліографічний список використаної літератури. Складає 4 сторінки;

– Додатки – містить технічне завдання і робочий проект.

Кількість рисунків: 9

ЗМІСТ

ВСТУП

Суть розробки проекту полягає у створенні програмного забезпечення для автоматизованого проектування силових трансформаторів. Програма передбачає у собі наявність певних інструментів для автоматизованого проектування силових трансформаторів.

За допомогою програмного забезпечення, що розроблюється, можна буде здійснювати автоматизоване проектування силових трансформаторів. Також у даній програмі користувач матиме можливість аналізувати показники у вигляді таблиці та графіку. Що дозволяє користувачу аналізувати які процеси використовують динамічну пам’ять.

Актуальність роботи: на сьогоднішній день дана робота є досить актуальною, оскільки автоматизоване проектування силових трансформаторів дозволяє оптимізувати та прискорити процес роботи. Велика кількість розробників та компаній зараз використовує різні програми для автоматизованого проектування силових трансформаторів. Тому, розробка даного проекту допоможе користувачам працювати більш ефективно та продуктивно. Також під час розробки даного проекту, було розглянуто переваги та недоліки існуючих аналогів та проведено опитування зацікавлених сторін, завдяки чому було покращено кінцевий продукт.

РОЗДІЛ 1 ЗБІР ТА АНАЛІЗ ВИМОГ ДО ЗАСТОСУНКУ

1.1 Огляд аналогів

В ході розробки програми дипломної роботи, було розглянуто аналоги програм для автоматизованого проектування силових трансформаторів, визначено їх переваги та недоліки, виконано покращення власної програми завдяки висновкам, зроблених під час огляду аналогів. Основна частина – це десктопні додатки. Частина функціоналу безкоштовна. Проте, для користування повним функціоналом потрібно оформити платну підписку. До уваги обрано найпопулярніші продукти ринку.

Процес проектування трансформатора супроводжується суперечливими вимогами до його властивостей. Це зумовлено тим, що трансформатор включає велику кількість елементів і виготовляється на обладнанні, що індивідуально настроюється. Водночас виробництво трансформаторів пов'язане з використанням таких цінних матеріалів як мідь, алюміній, сталь та ін. Отже, трансформатор є складною системою, якісне проектування якої важко без автоматизації. Пропонується прискорити процес розрахунку внутрішньої ізоляції трансформатора за допомогою паралельного обчислювального середовища та застосування швидких алгоритмів.

При проектуванні об'єктів електроенергетики на проектувальника одночасно накладаються вимоги до якості проекту та стислий термін його виконання. Тому рутинні розрахунки, що повторюються, раціонально автоматизувати і довірити програмним комплексам. У зв'язку з цим в електроенергетиці широко впроваджуються різні САПР, що полегшують виконання проектів та різних їх етапів. Впровадження САПР починалося з автоматизації виконання електротехнічних креслень за допомогою застосування спеціалізованих програмних пакетів [1, 2], які пропонують користувачеві бібліотеки елементів електричних схем, виконані відповідно до вимог ЕСКД. Подальшим розвитком САПР об'єктів електроенергетики було виконання розрахунків, що полегшують окремі етапи проектування. Так, для розрахунку світлотехнічної частини одним із найпоширеніших програмних пакетів є DIALux [3], розроблений

німецьким інститутом прикладної світлотехніки -DIAL GmbH (Deutehe Institut fur Angewandte Lichttechnik). САПР також використовуються для вирішення задач оптимізації [4-11]. У таких роботах, як [7] та [8] розглядаються особливості автоматизації проектування систем електропостачання сільськогосподарських споживачів. Автори [7] розробили програмний комплекс, що дозволяє знизити питому витрату палива. В [11] пропонується автоматизована система забезпечення надійності та якості обладнання ASONIKA, призначена для скорочення часу, що витрачається інженерами на цю роботу, а також зменшення кількості дефектів та витрат виробництва. САПР широко застосовуються для навчальних цілей [12-16]. Наприклад, у [12] викладено принципи побудови навчальної САПР електричної частини станцій та підстанцій, що базується на використанні стандартної системи проектування AutoCAD. Описано порядок розрахунку струмів КЗ та вибору обладнання та струмопроводів, покладений в основу алгоритмів роботи САПР. Ряд програм [17-21] дозволяють вибирати електрообладнання. Основним завданням у програмі, описаній у [20], є розрахунок потужності трансформатора та вибір перерізів кабелів на основі вихідних даних, заданих проектувальником відповідно до технічного завдання. [17] описаний програмний комплекс для вибору площі поперечних перерізів провідників за методом економічної щільності струму, а також вибору номінальної потужності трансформаторів за розрахунковою потужністю навантаження вузла. Найбільш популярні САПР для проектування електропостачання різних галузей [12-13]. Роботи [13-17] присвячені системам автоматизованого проектування електропостачання різних видів транспортних засобів. Ряд робіт спрямовані на покращення алгоритмів функціонування САПР та підвищення ефективності взаємодії інженера-проектувальника з автоматизованою системою [31-33]. Розглянуті САПР полегшують завдання інженерів-проектувальників тим, що дозволяють здійснювати проектування різних електроустановок чи його схем, і навіть автоматизувати процес виконання окремих етапів проекту. При аналізі опублікованих робіт і програмних продуктів для ЕОМ у галузі проектування об'єктів електроенергетики можна дійти невтішного висновку, що жодна з програм не дозволяє здійснити комплексне проектування розподільних пристроїв підстанції. У цій роботі авторами запропоновано

алгоритм автоматизованого розрахунку складових струму короткого замикання, а також вибору та перевірки високовольтних вимикачів. Запропонований алгоритм входить до складу оригінальної САПР [34], що реалізує комплексний підхід до автоматизованого виконання проекту підстанції та дозволяє на основі технічного завдання отримати комплект робочої документації, в яку проектувальник може вносити зміни на будь-якому етапі проектування.

1.2 Аналіз коментарів користувачів

Опитування проводилося серед розробників та пересічних користувачів. Інструмент опитування: google-форми. Для зацікавлених сторін було представлено основні питання, які допомогли би під час подальшого проектування програмного забезпечення.

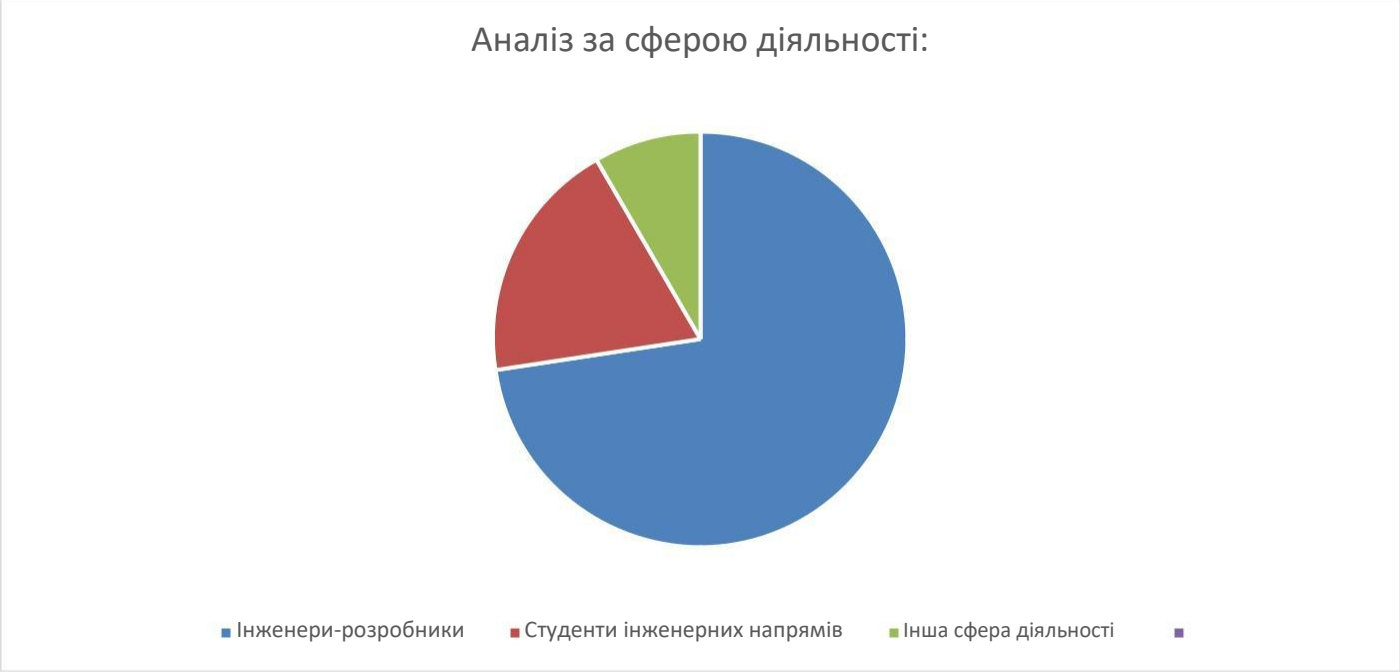


Рисунок 1.1 – Аналіз за сферою діяльності

Серед запитань опитування: проектування об'єктів електроенергетики, дотримання вимог якості проекту, терміни його виконання, полегшення виконання проектів, кількість етапів, наявність бібліотек, виконання розрахунків та ін.

1.3 Постановка задачі

Першочерговим завданням проекту є визначення загальних переваг та недоліків основних аналогів на ринку, а також необхідно зробити висновки стосовно покращення власного проекту.

Після цього провести опитування зацікавлених сторін та підкреслити для себе найголовніші потреби користувачів.

На основі потреб користувачів розробити власний додаток з алгоритмом, що реалізує комплексний підхід до автоматизованого виконання проекту підстанції та дозволяє на основі технічного завдання отримати комплект робочої документації, в яку проектувальник може вносити зміни на будь-якому етапі проектування. Встановлено основні етапи проектування:

1. Встановлення початкових значень ряду змінних величин. Обмотки ПН та ВН мають конструктивне виконання у вигляді багатошарового (одношарового) концентру безвертикальні канали. Для запуску ітераційного процесу приймається початкове значення ширини обмотки ПН, потім це значення уточнюється.

2. Розрахунок перерізу витків обмоток ПН та ВН.

3. Визначення висоти обмотки Δ.

4.Визначення ширини обмотки ВН:

5. Розрахунок напруги короткого замикання.

Після визначення напруги короткого замикання здійснюється його двостороння перевірка та регулювання.

6. Вибір типу обмотки ПН.

Висновки до розділу 1

Аналіз та збір вимог під час проектування програмного забезпечення є важливим етапом при розробці. Це допомагає спланувати подальшу роботу. За допомогою аналізу вимог можливо побачити основні недоліки аналогів ринку, що дає можливість виправити ці недоліки у своєму проекті.

Було опитано зацікавлені сторони. Зібрано статистику основних потреб користувачів, а також зібрано інформацію про досвід використання схожих (аналогічних) програм. Найбільш популярні САПР для проектування електропостачання різних галузей. Розглянуті САПР полегшують завдання інженерів-проектувальників тим, що дозволяють здійснювати проектування різних електроустановок чи його схем, і навіть автоматизувати процес виконання окремих етапів проекту. При аналізі опублікованих робіт і програмних продуктів для ЕОМ у галузі проектування об'єктів електроенергетики можна дійти невтішного висновку, що жодна з програм не дозволяє здійснити комплексне проектування розподільних пристроїв підстанції. У цій роботі авторами запропоновано алгоритм автоматизованого розрахунку складових струму короткого замикання, а також вибору та перевірки високовольтних вимикачів.

Використовуючи отриману інформацію, створено план подальшої роботи з урахуванням того, що запроектований алгоритм повинен реалізувати комплексний підхід до автоматизованого виконання проекту підстанції та дозволяє на основі технічного завдання отримати комплект робочої документації, в яку проектувальник може вносити зміни на будь-якому етапі проектування.

РОЗДІЛ 2 ПРОЕКТУВАННЯ

ЗАСТОСУНКУ 2.1 Задачі проекту

Початковим етапом при проектуванні підстанції є розрахунок струмів короткого замикання для перевірки електричних апаратів та провідників за умовами термічної та електродинамічної стійкості. У цій роботі пропонується алгоритм автоматизованого розрахунку складових струму короткого замикання, що враховує клас напруги розподільного пристрою, параметри джерела, спосіб зв'язку розподільного пристрою з джерелом нескінченної потужності. Алгоритм розрахунку складових струму короткого замикання заснований на діючих керівних вказівках [5]. Для розрахунку струмів КЗ необхідно знати найбільше початкове значення значення періодичної складової струму

2.2 Функціональні вимоги

При розробці алгоритму перевірки високовольтних вимикачів потрібно враховувати згасання аперіодичної складової струму короткого замикання. Для цього методом найменших квадратів визначити коефіцієнти регресійного рівняння, що описують залежність нормованої асимметрії струму, що відключається від часу розходження контактів вимикача. Дане регресійне рівняння оцінює нормовану асиметрію з похибкою, що не перевищує 5%. Потрібно розробити алгоритм, що дозволяє на основі даних про величину періодичності складової струму короткого замикання на шинах підстанції, класі напруги її розподільних пристроїв і потужності силових трансформаторів здійснити вибір і перевірку вимикачів і що відрізняється можливістю експорту номінальних параметрів електрообладнання з вбудованої в САПР бази даних. Даний алгоритм реалізований в САПР низьких підстанцій, дозволить автоматизувати виконання проекту електричної частини низьких підстанцій та на основі технічного завдання отримати комплект проектної документації з техніко-економічним обґрунтуванням прийнятих рішень.

2.3 Вхідні та вихідні дані

Початковим етапом при проектуванні підстанції є розрахунок струмів короткого замиканняня для перевірки електричних апаратів та проводників за умовами термічної та електродинамічної стійкості.

Пропонується алгоритм автоматизованого розрахунку зі що становлять струму короткого замикання, враховують клас напруги розподільчого пристрої, параметри джерела, спосіб зв'язку розподільного пристрою з джерелом без кінцевої потужності. Алгоритм розрахунку струму короткого замикання заснований на чинних керівних вказівках [35].

Відповідно до виокремлених етапів проектування необхідні наступні вхідні дані На етапі встановлення початкових значень ряду змінних величин. Вважається, що обмотки ПН та ВН мають конструктивне виконання у вигляді багатошарового (одношарового) концентру безвертикальні канали. Для запуску ітераційного процесу приймається початкове значення ширини обмотки ПН, наприклад а1=40мм. Потім це значення уточнюється.

На етупі розрахунку перерізів витків обмоток ПН та ВН (мм²) та на етапі визначенні обмотки:

W, - число витків обмотки НН:

Δа1, і ΔН1 - сумарні величини вертикальних та горизонтальних охолоджуючих каналів в обмотці ПН (для початкового етапу приймаються рівними нулю) відповідно; $б_{из1}$ - товщина міжшарової ізоляції; коефіцієнти в чисельнику враховують товщину ізоляції, нещільність намотування, усадку ізоляції тощо.

На етапі визначення ширини обмотки ВН (мм):

$Δа_2$, і $ΔН_2$ - сумарні величини вертикальних та горизонтальних охолоджуючих каналів В обмотці ВН (для початкового етапу приймаються рівними нулю) відповідно;

$б_{из2}$ - Товщина міжшарової ізоляції.

На етапі розрахунку напруги короткого замикання (%):

f - частота мережі;

$D_{ср} = d_c+2a_{01}+2a_1+a_{12}$ – середній діаметр каналу розсіювання, мм;

$т = (a_1+a_2) /3+a_1$ - ширина наведеного каналу розсіювання,

мм; $p_р = 1-(a_1+a_2+a_{12})/(\pi H1)$ Коефіцієнт Роговського;

$E_в$ - ЕРС витка;

коефіцієнт 1,02 враховує додаткове

розсіювання. На етапі проведення вибору типу

обмотки ПН. НН - гвинтова, безперервна та

циліндрична; ВН - безперервна та циліндрична.

Безперервна обмотка може застосовуватись у широкому діапазоні потужностей та напруг. Однак за малої кількості витків в обмотці ПН (менше 100-120) застосування безперервного типу обмотки стає або невиправданим, або неможливим. Перевагою безперервної обмотки є висока механічна міцність та зручність виконання регулювальних відгалужень; недоліком – підвищена складність виготовлення.

Інформація про тип силового трансформатора і номінальну напругу розподільного пристрою вищої напруги вводиться як вихідні дані проектувальником через інтерфейс користувача САПР.

Програма повинна забезпечити аналіз отриманих даних, а також, роботу в декількох режимах.

В програмі необхідно реалізувати зберігання зібраних даних.

Вхідні дані:

- Тип трансформатору.
- Потужність трансформатору.
- Частота мережі.
- Лінійна напруга обмотки НН.
- Лінійна напруга обмотки ВН.
- Схема з'єднання обмотки НН.
- Схема з'єднання обмотки ВН.
- Число ступенів регулювання в ВН.

- Відсоток регулювання у щаблі.
- Індукція.

Вихідні дані:

- Діаметр описаної окружності стержня магнітопроводу.
- Активний перетин сталі.
- Кількість витків в обмотці.
- Значення ЕДС витку.
- Максимальна кількість витків в обмотці.
- Висота обмотки.
- Ширина обмотки.
- Напруга короткого замикання.
- Перевищення температури обмотки.
- Додаткові втрати в обмотці.
- Критична напруга.
- Щільність теплового потоку.

Самі записи зберігаються у базі даних.
Вихідні дані виводяться у вигляді текстового файлу.

2.4 Вибір мови програмування

У наш час дуже багато мов програмування, але по-справжньому популярними і широко затребуваниє ми тільки деякі з них.

На сьогоднішній день існує більше 2 тисяч мов програмування, та постійно з'являються нові. У список найпопулярніших і використовуваних (з різних точок зору) входять близько 30-40 мов. Про склад цього списку можна судити за рейтингами популярності, які постійно змінюються і коригуються. Найчастіше це обумовлюється появою нових технологій, нових середовищ розробки програмного забезпечення та ін. Наприклад, можна назвати: Java, C, Objective-C, C ++, PHP, Python, Visual Basic, Perl, Ruby, JavaScript, Lisp, Pascal, Delphi/Object Pascal, SQL [3].

Мова програмування високого рівня якою ми будемо писати - це мова програмування, розроблена для швидкості і зручності використання програмістом. Основна риса високорівневих мов - це введення смислових конструкцій, що стисло описують такі структури даних і операції над ними, описи яких на машинному коді або іншою низькорівневою мовою програмування дуже довгі і складні для розуміння.

Для розробки програми за допомогою якої можливо аналізувати використання динамічної пам’яті, була обрана мова програмування С++.

С ++ - об'єктно-орієнтована мова програмування, що відповідає стандартам ANSI і Міжнародної організації стандартів (International Standards Organization - ISO). Об'єктна орієнтованість С ++ означає, що вона підтримує стиль програмування, що спрощує кодування великомасштабних програм і забезпечує їх розширюваність. С ++ може генерувати досить ефективні, високошвидкісні програми. Сертифікація ANSI і ISO забезпечила переносимість С ++, написані на С++ програми сумісні з більшістю сучасних середовищ програмування [14].

Головною особливістю є набір визначених класів, типи даних яких можуть бути створені кілька разів. Класи: можна додатково розмістити членів функції для реалізації

певних функцій. Кілька об'єктів певного класу можуть бути визначені для реалізації функцій в класі. Об'єкти можуть бути визначені як екземпляри, створених під час виконання. Ці класи також можуть бути успадковані від інших нових класів [14].

2.5 Опис компонентно-орієнтованого підходу

Парадигма об'єктно-орієнтованого програмування передбачає, що будь-яка програмна система проектується як сукупність об'єктів. Зазвичай у літературі з об'єктно-орієнтованих мов описуються основні конструкції мов та приклади їх використання.

Як правило, використання ізольованих об'єктів не дозволяє досягти поставленої мети. Інтересом є взаємодія об'єктів, тобто обмін інформацією між об'єктами. Зазвичай стверджують, що один об'єкт взаємодіє з іншим у вигляді повідомлень, тобто у процесі взаємодії об'єкти обмінюються повідомленнями. Повідомлення – це виклик методу. Спільне використання методів різних об'єктів - показник взаємодії об'єктів. Отже, методи одного об'єкта повинні викликати методи інших об'єктів. І тут жоден метод, що у взаємодії, неспроможний досягти мети самостійно. Саме тому важливо правильно організувати обмін інформацією між методами різних об'єктів. Такий обмін передбачає, що результати роботи методу одного об'єкта є вихідними даними методу іншого об'єкта.

Об'єктно-орієнтоване програмування (далі ООП) є одним із методів програмування, ідея якого була заснована на представлення програми, як сукупність об'єктів. Кожен об'єкт є екземпляром класу, з яким він пов'язаний. Класи ж у свою чергу утворюють успадковану ієрархію.

Якщо глянути з ідеологічного погляду, то ООП більше нагадуватиме моделювання інформаційних об'єктів, оскільки класи є відображення реальних сутностей. ООП стало замінювати структурне програмування, шляхом болю раціонального вирішення його основного завдання: структурування даних з погляду управління ними. При реалізації великих проектів це буває надто необхідно, оскільки це покращує процес управління та процес моделювання.

В даний час об'єктно-орієнтований підхід при розробці систем різного ступеня складності є загальновизнаним [5, с. 3]. Понад те, він застосовується як під час розробки, а й під час використання широко поширених объектно-ориентированных систем.

Нестача процедурно-орієнтованих мов полягає в наявності двох проблем: той факт, що поділ даних і функцій, що є основою структурного підходу, погано відображає картину реального світу.

Під ООП розуміється технологія, що з'явилася як реакція на чергову фазу кризи програмного забезпечення, коли методи структурного програмування вже не дозволяли справлятися зі складністю промислових програмних продуктів, що зростає.

2.6 Проектування екранів прототипу застосунку

Застосунок повинен мати власну зручну структуру, зрозумілу на інтуїтивному рівні, яка просто і невимушено розкладає всю інформацію «по поличках». Важливо цінувати час користувача. Адже, в кінцевому підсумку, саме від нього залежить успіх проекту в цілому.

До дизайну належить не тільки зовнішній вигляд застосунку, але і його функціональне наповнення, структура і навігація.

UX – це про допомогу користувачу в рішенні задачі, про логіку роботи застосунку, про емоції задоволення від роботи з інтерфейсом.

«UX» перекладається як «користувацький досвід». Призначений для користувача досвід застосування визначається тим, як користувачі взаємодіють з застосунком. Чи є досвід користувача інтуїтивно-зрозумілим або навпаки заплутує його? Навігація застосунку здається логічною або не піддається логіці? Чи дає людям взаємодія з застосунком відчуття, що вони ефективно виконують завдання, які вони поставили перед собою, або це схоже на блукання по замкненому колу і відчуття безвиході?

Головним завданням створення UX-концепції розробки застосунку було визначення оптимальної структури інтерфейсу і його функціональності, а також організація основних функціональних блоків інтерфейсу, пов'язаних один з одним. Тому

що, якщо інтерфейс буде добре працювати, у користувача буде хороший досвід після взаємодії з таким інтерфейсом.

Але якщо навігація складна або інтуїтивно незрозуміла, то швидше за все, користувач зазнає негативного досвіду при взаємодії з інтерфейсом. Таким чином, основне завдання UX дизайну полягає в тому, щоб уникнути другого сценарію.

Досвід користувача визначається тим, наскільки легко чи складно користувачу взаємодіяти з елементами інтерфейсу, створеного дизайнером.

Було розроблено функціональну структуру застосунку. Було вирішено, що головна сторінка застосунку буде побудована стандартним чином і матиме основний екран.

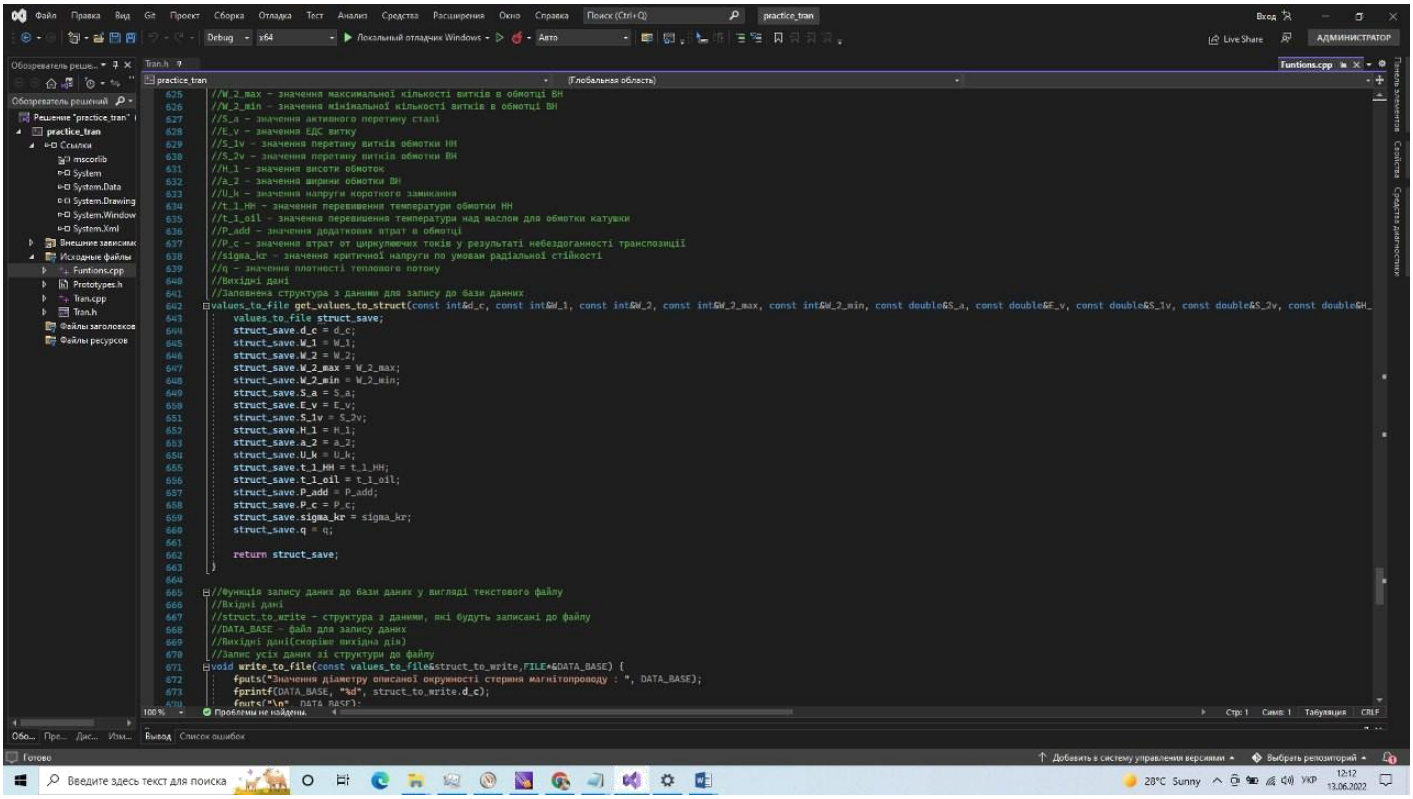
Стосовно навігації застосунку, застосунок має меню з рубриками, кнопки та панелі управління. Завдяки компактному розташуванню основних функціональних блоків, користувачеві не доводиться нескінченно шукати потрібні елементи.

2.7 Проектування навігації між екранами прототипу застосунку

UI – дизайн охоплює не тільки графічний інтерфейс, а ще й тактильний, голосовий або звуковий. Це процес втілення в візуальних деталях користувацького досвіду, який розробили в UX-дизайні на підставі вивчення клієнта, дослідження цільової аудиторії. UI-дизайн включає в себе етапи робіт над візуальної або графічної частиною інтерфейсу: кнопками, меню, іконками, анімацією, ілюстраціями, меню, колажами та шрифтами.

2.8 Проектування дизайну інтерфейсу прототипу застосунку

Дизайн – це дуже важлива частина запуску застосунку, він повинен допомагати, а не заважати сприймати інформацію. Але, тим не менш, дизайн повинен розповісти або натякнути на те, про що саме застосунок.



```
//W_2_max – значення максимальної кількості витків в обмотці ВН
//W_2_min – значення мінімальної кількості витків в обмотці ВН
//S_a – значення активного перетину сталі
//E_v – значення ЕДС витку
//S_1v – значення перетину витків обмотки НН
//S_2v – значення перетину витків обмотки ВН
//H_1 – значення висоти обмоток
//a_2 – значення вирижи обмотки ВН
//U_k – значення напруги короткого замикання
//t_1_HH – значення перевищення температури обмотки НН
//t_1_oil – значення перевищення температури над маслом для обмотки катушки
//P_add – значення додаткових втрат в обмотці
//P_c – значення втрат от циркулюючих токів у результаті небездоганності транспозиції
//sigma_kr – значення критичної напруги по умовам радіальної стійкості
//q – значення платності теплового потоку
//Вихідні дані
//Заповнена структура з даними для запису до бази данних
values_to_file get_values_to_struct(const int&d_c, const int&W_1, const int&W_2, const int&W_2_max, const int&W_2_min, const double&S_a, const double&E_v, const double&S_1v, const double&S_2v, const double&H_
    values_to_file struct_save;
    struct_save.d_c = d_c;
    struct_save.W_1 = W_1;
    struct_save.W_2 = W_2;
    struct_save.W_2_max = W_2_max;
    struct_save.W_2_min = W_2_min;
    struct_save.S_a = S_a;
    struct_save.E_v = E_v;
    struct_save.S_1v = S_2v;
    struct_save.H_1 = H_1;
    struct_save.a_2 = a_2;
    struct_save.U_k = U_k;
    struct_save.t_1_HH = t_1_HH;
    struct_save.t_1_oil = t_1_oil;
    struct_save.P_add = P_add;
    struct_save.P_c = P_c;
    struct_save.sigma_kr = sigma_kr;
    struct_save.q = q;

    return struct_save;
}

//функція запису даних до бази даних у вигляді текстового файлу
//Вхідні дані
//struct_to_write – структура з даними, які будуть записані до файлу
//DATA_BASE – файл для запису даних
//Вихідні дані(скоріше вихідна дія)
//Запис усіх даних зі структури до файлу
void write_to_file(const values_to_file&struct_to_write,FILE*&DATA_BASE) {
    fputs("Значення діаметру описаної окружності стержня магнітопроводу : ", DATA_BASE);
    fprintf(DATA_BASE, "%d", struct_to_write.d_c);
    fputs("\n", DATA_BASE);
```

Рис. 2.1. - Скріншот застосунку

У ньому повинні бути розставлені всі необхідні аспекти:

- якісний, яскравий логотип, що легко сприймається і запам'ятовується, за яким можна визначити вид діяльності компанії;

- колір, відповідний образу, тематичним напрямком і динаміки застосунку;

- шрифт застосунку.

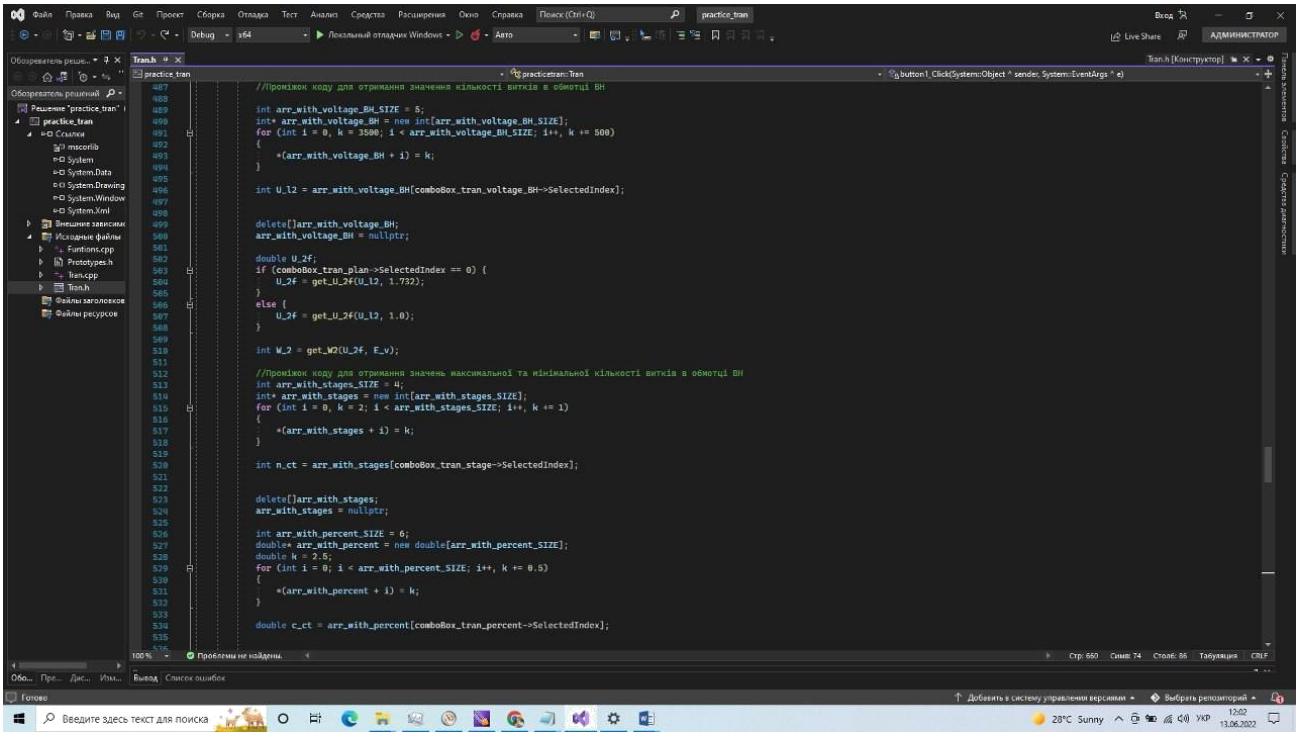


Рис. 2.2. - Скріншот застосунку

Дуже важливо не забувати про те, що якщо дизайн застосунку красивий, але через нього важко прочитати інформацію, з ним важко працювати та виконувати функціональні завдання, то на ньому ніхто не затримується і просто змінить його, наприклад, на застосунок конкурентів.

РОЗДІЛ 3 РОЗРОБКА ЗАСТОСУНКУ

3.1 Розробка системи

1.Встановлюються початкові значення ряду змінних величин. Вважається, що обмотки ПН та ВН мають конструктивне виконання у вигляді багатошарового (одношарового) концентру безвертикальні канали. Для запуску ітераційного процесу приймається початкове значення ширини обмотки ПН, наприклад а1=40мм. Потім це значення уточнюється.

2. Розраховується переріз витків обмоток ПН та ВН (мм²):

```
//Функція отримання значення перетину проводу
//Вхідні дані
//a_pr - значення ширини проводу
//b_pr - значення висоти проводу
//Вихідні дані
//Отримане значення перетину проводу округлене до тисячних
double get_S(const double& a_pr, const double& b_pr) {

    return round((0.98 * a_pr * b_pr) * 1000) / 1000;

}

//Функція отримання значення перетину витку
//Вхідні дані
//x - 1
//S - значення перетину проводу
//Вихідні дані
//Отримане значення перетину витку округлене до тисячних
double get_S_v(const int& x, const double& S) {

    return round((S * x) * 1000) / 1000;
```

Рис. 3.1. Вікно написання коду

$$S_{1в} = I_{1ф}/\delta_1; \quad S_{2в} = I_{2ф}/\delta_2;$$

3. Визначається висота обмотки (мм):Δ

```
//Функція отримання значення висоти обмоток
//Вхідні дані
//S_1v - значення перетину обмотки НН
//W_1 - значення кількості витків в обмотці НН
//b_iz1 - 1 товщина міжслоїовой ізоляції, константа зі значенням 0.35
//Вихідні дані
//Отримане значення висоти обмоток округлене до тисячних
double get_H1(const double& S_1v, const int& W_1, const double& b_iz1) {

    return round((1.05 * 1.1 * S_1v * W_1 / b_iz1) * 1000) / 1000;

}
```

Рис. 3.2. Вікно написання коду

$$H_1 = [1{,}05 \text{ x } 1{,}1\ S_{1В} W_1/(a_1 - \Delta a_1 - b_{из1}] + \Delta H1,$$

де W, - число витків обмотки НН: Δа1, і ΔН1 - сумарні величини вертикальних та горизонтальних охолоджуючих каналів в обмотці ПН (для початкового етапу приймаються рівними нулю) відповідно; биз1 - товщина міжшарової ізоляції; коефіцієнти в чисельнику враховують товщину ізоляції, нещільність намотування, усадку ізоляції тощо.

4.Визначається ширина обмотки ВН (мм):

```
//Функція отримання значення ширини обмотки ВН
//Вхідні дані
//S_2v - значення перетину обмотки ВН
//W_2_max - максимальне значення кількості витків в обмотці НН
//H_1 - значення висоти обмоток
//b_iz2 - значення 2 товщини ізоляціїної обмотки
//Вихідні дані
//Отримане значення ширини обмотки ВН округлене до тисячних
double get_a_2(const double& S_2v, const int& W_2_max, const double& H_1, const double& b_iz2) {

    return round(((1.05 * 1.1 * S_2v * W_2_max / H_1) + b_iz2) * 1000) / 1000;

}
```

Рис. 3.3. Вікно написання коду

$$a_2 = [1{,}05 \text{ x } 1{,}1\ S_{2В} W_{2max}/(H_1 - \Delta H_2)] + \Delta a_2 + b_{из2},$$

Де Δa_2, і ΔH_2 - сумарні величини вертикальних та горизонтальних охолоджуючих каналів В обмотці ВН (для початкового етапу приймаються рівними нулю) відповідно; $б_{из2}$ - Товщина міжшарової ізоляції.

Значення $б_{из1}$ і $б_{из2}$ розраховуються за емпіричними формулами $б_{из1} = 2 \times 0{,}12(8W1/H1-2)$, $б_{из2} = 4 \times 0{,}12(7W2max/H1 - 2)$. В тому випадку, якщо одне з значень $б_{из1}$ або $б_{из2}$ виходить негативним, приймається $б_{из}=0$.

5. Розраховується напруга короткого замикання (%):

```
//Функція отримання значення напруги короткого замикання
//Вхідні дані
//I_1f - значення 2 фазного току обмотки
//W_1 - значення кількості витків в обмотці НН
//f - частота сітки
//D_cp - значення середьного діматеру каналу розсіяння
//t - значення ширини приведенного каналу розсіювання
//P_p - значення коефіцієнту Роговського
//E_v - значення ЕДС витку
//H_1 - значення висоти обмоток
//Вихідні дані
//Отримане значення напруги короткого замикання округлене до тисячних
double get_U_k(const double& I_1f, const int& W_1, const int& f, const double& D_cp, const double& t, const double& P_p, const double& E_v, const double& H_1) {

    return -1 * round((I_1f * W_1 * f * D_cp * t * P_p * 1.02) / (40.5 * pow(10, 3) * E_v * H_1) * 1000) / 1000;

}
```

Рис. 3.4. Вікно написання коду

$$И_к = I_{1ф} W_1 f D_{ср} \tau \rho \times 1{,}02/(40{,}5 \times 10^3 E_в H_1),$$

Де f - частота мережі; $D_{ср} = d_c+2a_{01}+2a_1+a_{12}$ – середній діаметр каналу розсіювання, мм; $т = (a_1+a_2) /3+a_1$ - ширина наведеного каналу розсіювання, мм; $р_р = 1-(a_1+a_2+a_{12})/(\pi H1)$ Коефіцієнт Роговського; $Е_в$ - ЕРС витка; коефіцієнт 1,02 враховує додаткове розсіювання.

Після визначення напруги короткого замикання здійснюється його двостороння перевірка (рис. 2.8) та регулювання за рахунок зміни a_1. Кількість ітерацій при цьому не перевищує 2-5

6. Проводиться вибір типу обмотки ПН.

6. Проводиться вибір типу обмотки ПН.

Ця проектна операція може самостійно чи автоматично. У підсистемі передбачені розрахунок та розкладка наступних типів обмоток: НН - гвинтова, безперервна та циліндрична; ВН - безперервна та циліндрична.

Гвинтовий тип обмотки застосовується при великих значеннях струму в обмотці та невеликій кількості витків, що має місце при напрузі від 0,23 до 6,3 кВ. Наявність масляних каналів між сусідніми витками забезпечує необхідний тепловий режим обмотки цього.

Безперервна обмотка може застосовуватись у широкому діапазоні потужностей та напруг. Однак за малої кількості витків в обмотці ПН (менше 100-120) застосування безперервного типу обмотки стає або невиправданим, або неможливим. Перевагою безперервної обмотки є висока механічна міцність та зручність виконання регулювальних відгалужень; недоліком – підвищена складність виготовлення.

3.2 Розробка екранів застосунку

Функціональне призначення – програмний продукт являє собою програму розробки програмного забезпечення для автоматизованого проектування силових трансформаторів.

3.2 Розробка екранів застосунку

Функціональне призначення – програмний продукт являє собою програму розробки програмного забезпечення для автоматизованого проектування силових трансформаторів.

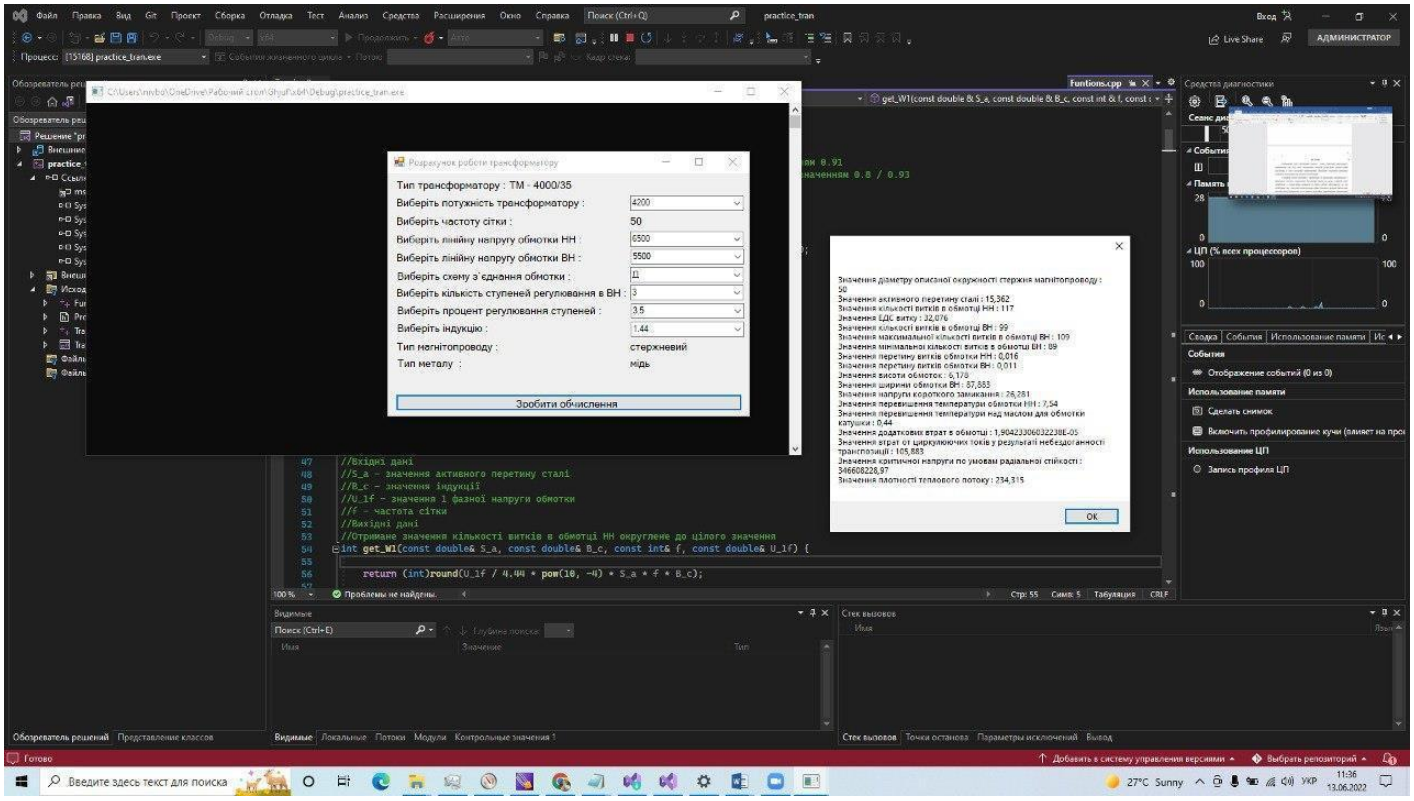


Рисунок 3.5 – Екранна форма застосунку

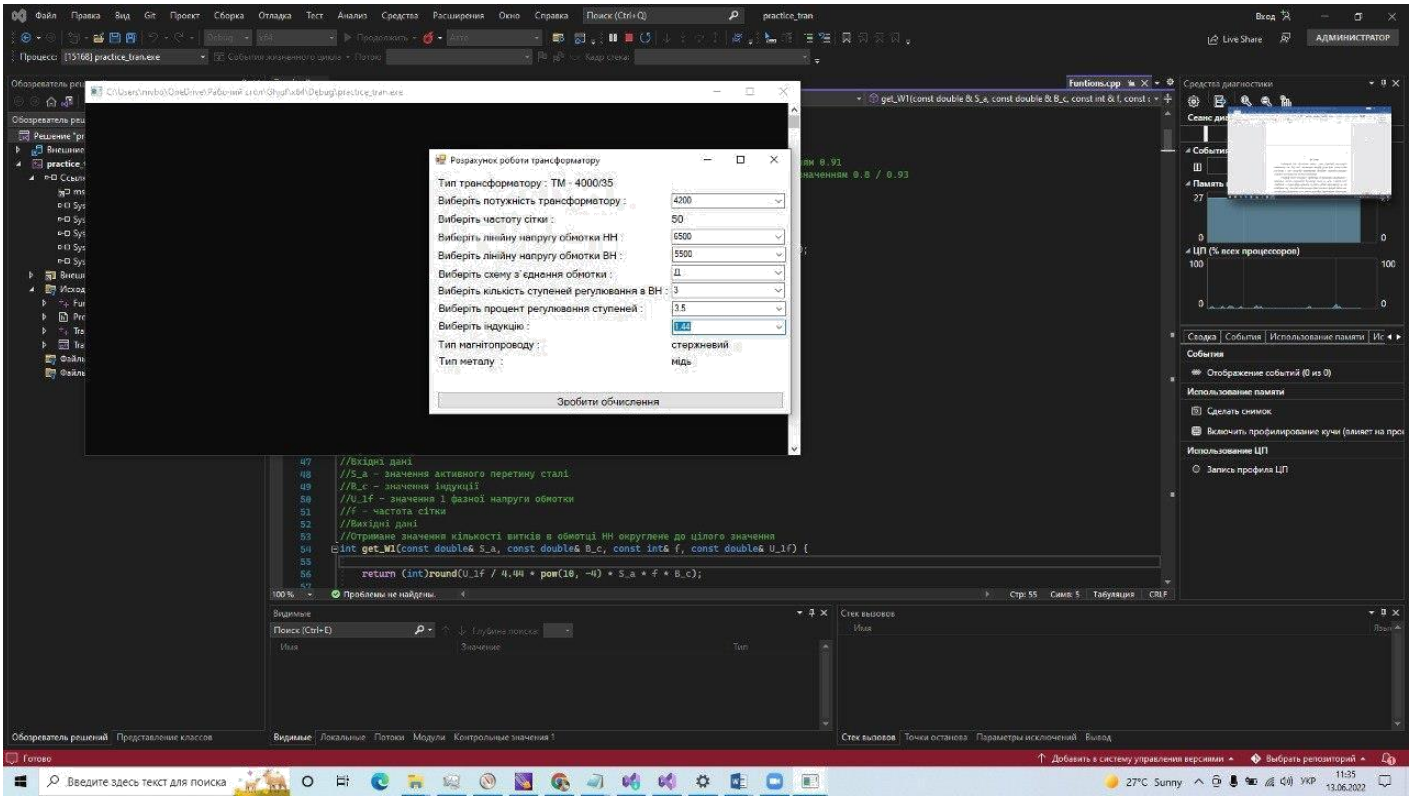


Рисунок 3.6 –Екранна форма введення вхідних данних

- Funtions.cpp – модуль призначений для розрахунку всіх функцій і запису їх у файл.
- Prototipes.h – модуль призначений для прототипування змінних
- Tran.cpp – модуль призначений для налагоджнгя логіки форм
- Tran.h – модуль призначений для налагодженя логіки елементів форм

3.3 Розробка інтерфейсу застосунку

Інтерфейс застосунку виконаний в зелено-синій кольоровій гамі, дизайн застосунку виконаний у стилі Material Design. Зелено-сині кольори відповідають основній тематиці застосунку. Основні елементи дизайну: кнопки, меню, панелі, та шрифти підібрані з урахуванням функціоналу, придатності для застосування користувачами.

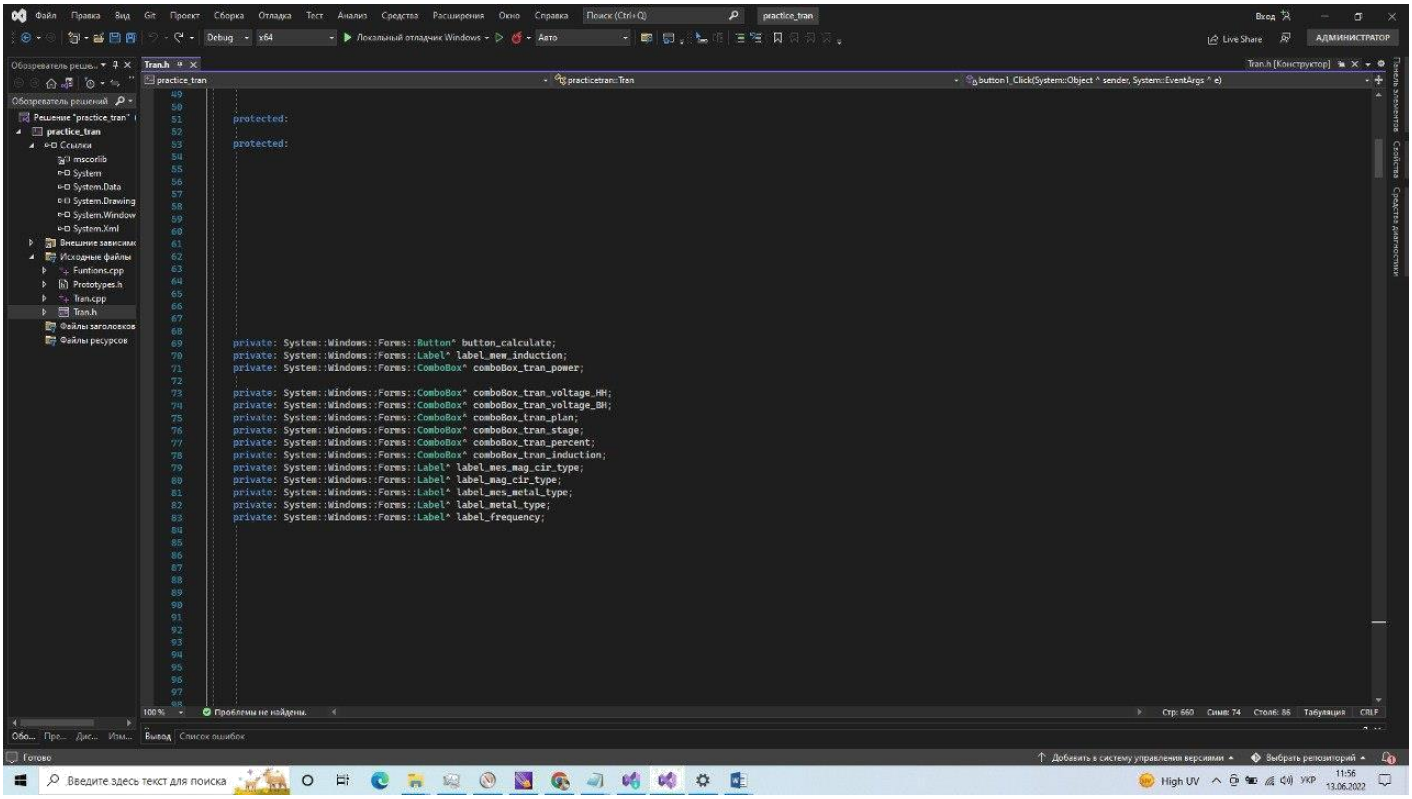


Рисунок 3.7 – Вікно написання коду

В UI-дизайні застосунку використані сучасні шрифти, підібрані аби відповідати дизайн-концепції сайту, мати вишукану, сувору, ділову форм.

Перед розробкою UI-концепції дизайна застосунку було досліджено сучасні тренди в галузі дизайну застосунків, нові ідеї, інструменти та технології в сфері дизайну.

Головним завданням розробки UI-дизайну – було створення простого, красивого дизайну графічного інтерфейсу проекту.

3.4 Розробка навігації застосунку

Необхідність у проектуванні архітектури системи була зумовлена тим, що у програмі реалізована велика кількість функціоналу. Від збирання даних до відображення їх у таблицях та діаграмах. Архітектура програми повинна включає в себе всі програмні модулі і компоненти та їх взаємодія між собою.

До переваг можливо віднести:

– Гарно продумана архітектура допомагає розширювати, змінювати та навіть налагоджувати та тестувати усю систему;

– Дає можливість змінювати архітектуру навіть старі версії;

– Підвищує швидкість розробки, особливо великого та важкого програмного забезпечення;

– Простіша реалізація порівняно з іншими підходами;

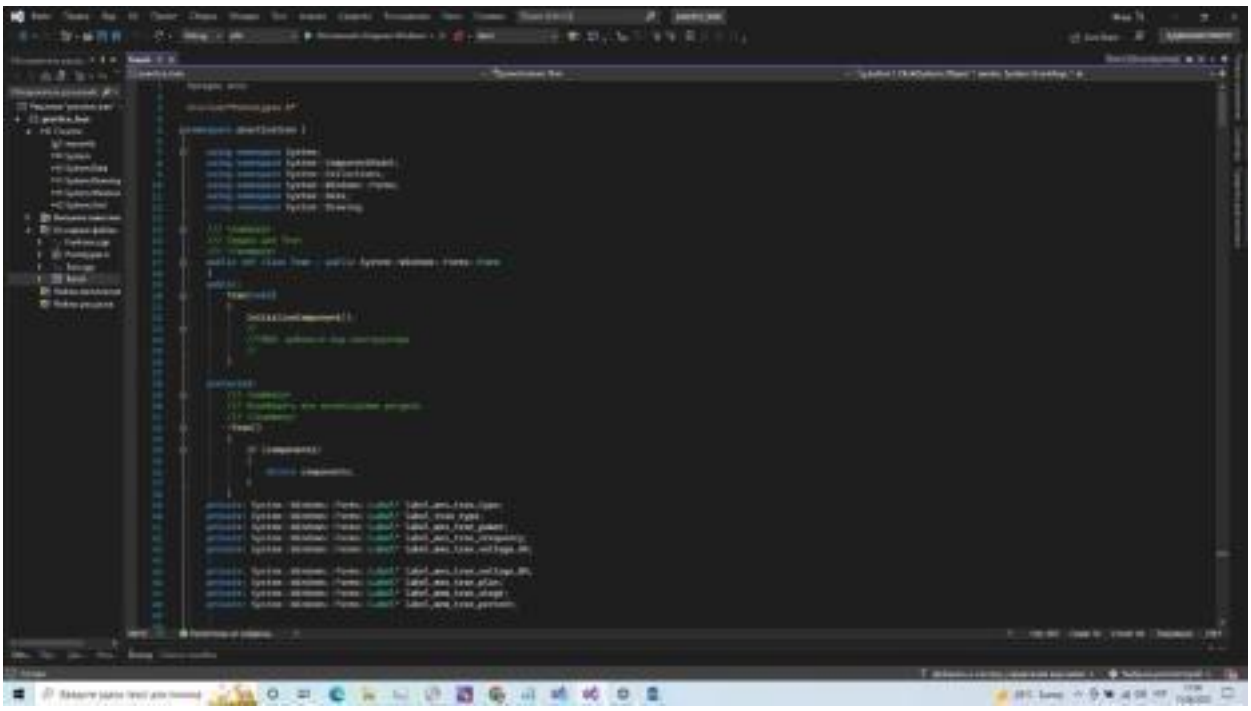


Рисунок 3.8 –Екрани застосунку

До недоліків можливо віднести:

– На реалізацію потрібно набагато більше часу;

– Потрібне комплексне тестування усієї системи;

– Важко продовжувати розробку програмного забезпечення після певного моменту, відбувається нагромадження усіх компонентів;

– Новому розробнику необхідно багато часу, щоб зрозуміти усю систему;

Проектування архітектури системи на пряму впливає на проектування інтерфейсу програмного забезпечення. Проектування інтерфейсу буде виділено в окремий розділ.

Оцінивши усі переваги та недоліки обраного підходу проектування архітектури системи можливо зробити висновки. Що для розробки програмного забезпечення цей підхід пасує як найліпше.

3.5 Розробка дизайну застосунку

Для створення інтерфейсу програми було проаналізовано аналоги, визначено їх певні переваги та недоліки. Зрозумівши як інтерфейси аналогічних програм проектували розробники у великих компаніях. Вирішено взяти прототип саме з працюючих програм та доробити його під сформовані вимоги.

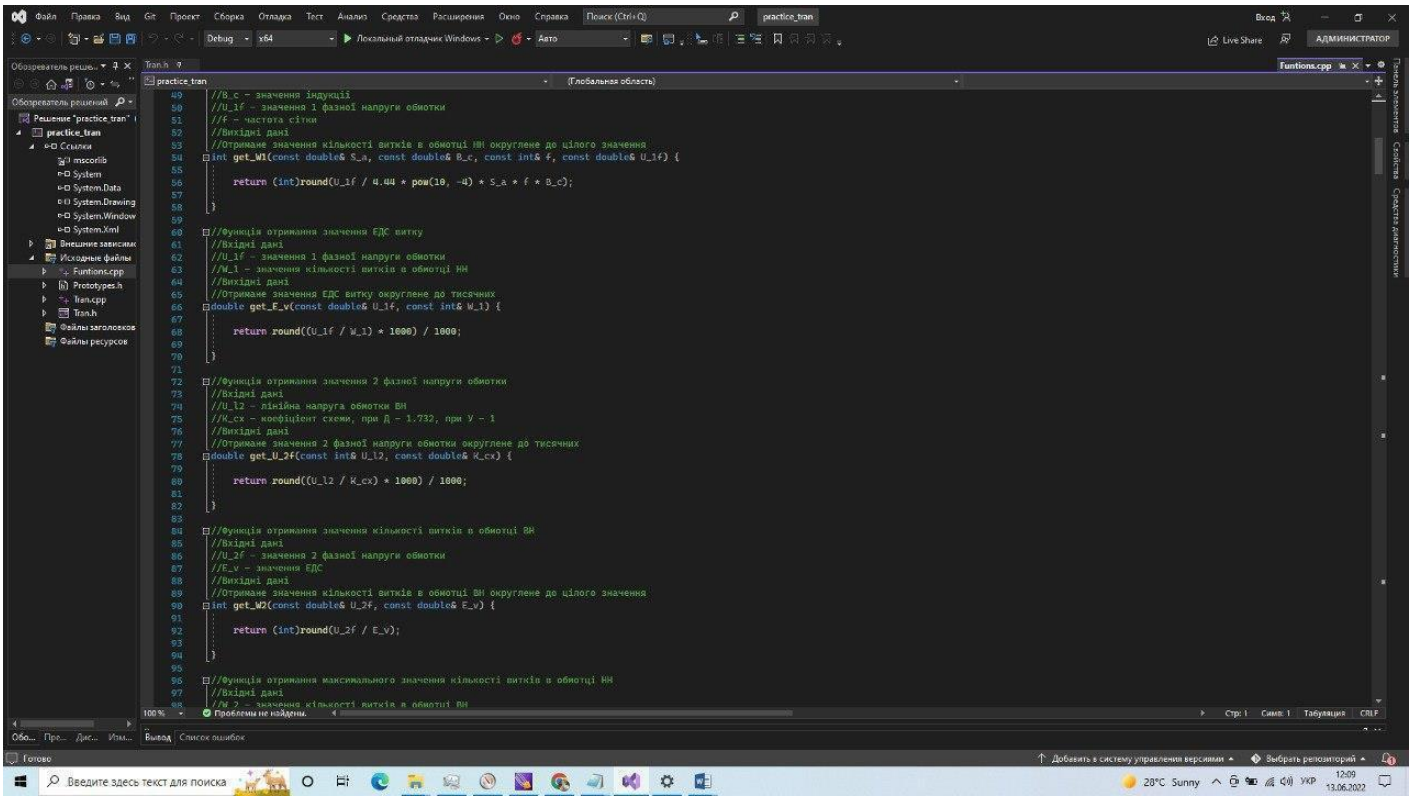


Рисунок 3.9 – Приклад роботи застосунку

В цілому застосунок забезпечений зручною і простою навігацією, зрозумілим UX-рішенням: відкрите меню з рубриками, система посилань, кнопок та ін.

Висновки до розділу 3

У цьому розділі було описано основні етапи розробки програмного забезпечення. А саме, формалізацію задачі, вибір мови програмування, описано усі важливі етапи проектування програми та етапи проектування інтерфейсу користувача. Були створені та описані макети програми для кращого розуміння програмного забезпечення. Також надано детальний опис усіх елементів інтерфейсу, надано пояснення яку інформацію та які дії він демонструє.

Описано функціональну структуру програми та її основні елементи. У результаті можливо зрозуміти усю структуру проекту та структуру інтерфейсу програми.

РОЗДІЛ 4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕПЕЧЕННЯ

4.1 Методи тестування програми

Тестування – один із найважливіших етапів розробки програмного продукту. Його необхідність полягає у аналізі програмного забезпечення на відповідність стандартним вимогам та у виявленні помилок у роботі продукту.

Безпосередньо від тестування залежить якість роботи програмного забезпечення, наскільки коректно воно працюватиме, його зручність та особливості користування. Завдяки тестуванню можна завчасно виявити помилки у роботі та виправити їх.

Серед варіантів тестування слід виділити структурне та статичне тестування. Данні типи перевірки відрізняються своєю зручністю, мають безліч переваг та ідеально підходять для випадків тестування програмного продукту, що досліджує використання динамічної пам’яті.

Особливості статичного тестування

Варіант статичного тестування дозволяє перевірити необхідні фрагменти коду та системні елементи ще до запуску програмного забезпечення. Це дозволяє побачити помилки та виправити якомога раніше, своєчасно коректує роботу системи та звільняє від незручностей при пошуку проблем у вже фактично працюючому забезпеченні. Проте, важливо виконати цей вид тестування у потрібний момент та на реальних пристроях. Щоб отримати точні результаті та побачити всі ймовірні похибки, слід проводити тестування на готовій до запуску системі.

Специфіка структурної перевірки

Стратегія структурного тестування базується на перевірці окремих компонентів системи. Це дозволяє виявити більш точну локалізацію помилки. Перевірка проводиться досвідченними тестувальниками, адже для реалізації таких випробувань необхідно добре розуміти саме програмне забезпечення, знати системні та апаратні вимоги до пристрою. Саме це можна вважати за недолік користування даної стратегії.

Проте, не дивлячись на це, така стратегія значно полегшує процес перевірки. Завдяки структурному тестуванню можна оптимізувати функціональну частину коду, відразу знайти та виправити помилки до передачі її кінцевому користувачу. Це значно економить час, хоча і потребує глибоких знань та розуміння тестованої системи
Саме ці популярні варіанти аналізу дозволяють вирішити проблеми з програмною системою на будь-якому етапі її створення та реалізації.

Налагодження роботи програмного забезпечення

Налагодження – це процес пошуку, виявлення та усунення помилок у роботі програмного забезпечення. Цей етап передбачає роботу над отриманими даними під час тестування, їх аналіз та постановка завдання над виправлення знайдених помилок. З ефективних методів слід виділити такі варіанти:

Ручне – коли знайденні помилки редагуються у ручному режимі в обраному «проблемному» локальному фрагменті.

Індукції – тут необхідно провести аналіз проблемного коду, аби виявити у яких випадках виникає помилка, від чого це залежить та підібрати варіанти її рішення.

Дедукції – відстежити всі можливі причини виникнення помилки та проаналізувати їх. Надалі відкинути всі неможливі варіанти, знову провести перевірку, проаналізувати проблему та знову відкинути недійсні варіанти, поки не залишиться одна з найвірогідніших гіпотез появи та рішення помилки.

Зворотнє відстеження – для тієї частини коду, де були виявленні проблеми висунути припущення про всі можливі значення змінних, які можуть вплинути на появу похибки. Виходячи з аналізу такої перевірки прорахувати значення цих змінних у попередніх точках.

Процес налагодження проводиться до тих пір, поки не буде виявлена та виправлена помилка.

4.2 Ручне тестування методами чорної та білої скриньки

Основними категоріями тестування слід вважати методи чорної та білої скриньки.Вони різняться між собою, тому потребують різного підходу до проектування програмного тесту. Завершення цих етапів гарантує отримання якісного продукту.

Метод чорної скриньки

Тестування відбувається без доступу фахівця до тексту програми. Тестувальник не знає внутрішню структуру програмного забезпечення від тестує, і працює виключно із зовнішнім інтерфейсом продукту.

Метод чорної скриньки застосовується на таких етапах тестування:

- Тестування системи;
- Тестування інтеграції;
- Приймальні випробування.

Кількість методів тестування залежить від складності продукту.

Для розробки алгоритмів тестування методом чорної скриньки використовують такі підходи:

- Причинно-наслідкові (визначення випадків та їх вплив на систему);
- Аналіз крайніх значень (визначення меж введення);
- Розмітка еквівалентності (дійсні та недійсні розмітки).

Під час аналізу не потрібно звертати увагу на архітектуру програми. Основна робота проходить з інтерфейсом забезпечення , де відомі вхідні та вихідні дані.

Згідно зі сертифікацією ISTQB:

Було проведено тестування без доступу до даних програми. Основна робота була з інтерфейсом програми, де відомі вхідні та вихідні дані. Інформація про обробку вхідних даних відсутня.

Метою є пошук помилок у таких категоріях:

- неправильно реалізовані або відсутні функції;

Всі функції присутні та реалізовані вірно.

– можливі помилки інтерфейсу;

Було виявлено некоректне відображення вікна з активними операціями.

– помилки поведінки або низька продуктивність програмного забезпечення;

Виявлено затримку у функціоналі програми при отриманні даних з використання динамічної пам’яті. Причиною стало велике навантаження на систему.

Метод білої скриньки

Основна мета цього аналізу – перевірка коду, його внутрішньої структури. Тестувальнику повністю доступний код, тому цей метод має іншу назву як «відкрите тестування». Процес тестування прозорий, є можливість перевірки вхідних та вихідних даних, є доступ до внутрішніх алгоритмів роботи системи.

Тестування білої скриньки передбачає пошук та покращення наступних моментів:

- Неробочих чи неоптимізованих ділянок коду;
- Проблеми з безпекою;
- Робочі процеси та сценарії введення;
- Умовні процеси;
- Неправильне функціонування об'єктів;
- Некоректне відображення інформації.

Головна мета тестування білої скриньки – підтвердження коректної працездатності програмного забезпечення. Будь-яка розбіжність з очікуваними результатами може бути класифікована як помилка, що потребує виправлення.

Згідно зі сертифікацією ISTQB:

Було проведено тестування з метою перевірки наявності та правильності введених даних, наявності чіткої логічної структури у всіх фрагментах коду, логічного шляху у програмному забезпеченні. Були встановлені логічні точки в різних

фрагментах програми для визначення статусу, чи відповідає фактичний стан продукту очікуваним показникам.

4.3 Аналіз помилок та оцінка роботи програмного забезпечення

Метою тестування програмного забезпечення був аналіз та пошук недоліків у фактично готовому системному продукту, а також підбір заходів щодо їх усунення. Були розглянуті методи покращення ефективності роботи програми, її точності та в цілому зручності використання.

В роботі були використані такі методи тестування:

- Модульний аналіз;

Проведення модульного аналізу дозволило створити якісний та оптимізований код. Під час тестування були перевірені фрагменти коду, їх послідовність та логічність написання. Завдяки аналізу були виявлені та виправлені помилки.

Типи нефункціонального тестування які використовувались:

- Продуктивність;

У результаті тестування вдалося максимально довершити роботу з точки зору стабільності та її високої продуктивності. Завдяки аналізу та роботі над помилками, зросла швидкість відгуку програмного забезпечення, введення та обробки даних, розрахунок результатів.

Використовуючи різні методи тестування, був проведений аналіз якості та правильності роботи програмного забезпечення, внесені виправлення щодо функціоналу системи, продуктивності роботи та зручності користування інтерфейсом.

Висновки до розділу 4

Даний розділ містить опис методів, аналіз проведеної роботи та результати тестування програмного забезпечення. Під час випробування системного продукту були виділені використанні стратегії для оптимізації коду, виправленні помилок та забезпеченні максимальної швидкості програми.

У розділі були розглянуті популярні методи тестування та налагодження програмного забезпечення, обґрунтування доцільності застосування у даній роботі, описано процес їх практичного використання та отримання результатів.

У процесі роботи були виявлені та виправлені недоліки програми, що дозволило підвищити швидкість виконання розрахункового процесу. Завдяки функціональному тестуванню були виявленні та виправленні похибки під час обробки вхідних даних.

Результатом тестування дизайну програмного продукту були виправлені деякі фрагменти, що могли вплинути на відображення певних елементів інтерфейсу.

ВИСНОВКИ

Результатом виконання бакалаврської роботи є десктопний застосунок в ОС Windows. Було проведено повний цикл створення програмного забезпечення для персональних комп’ютерів.

Було проведено аналіз існуючих сервісів у даній галузі, проаналізовано основні недоліки та переваги їх використання, досліджено особливості функціоналу, проведено аналіз UI/UX рішень.

На першому етапі було проведено огляд аналогів, опитування зацікавлених сторін, та постановка задачі. Цей етап був необхідний для формування орієнтирів у подальшій розробці програмного забезпечення.

На основі проведеного аналізу в ході виконання проєкту було запропоновано критерії оцінки програмних засобів з метою здійснення порівняльного аналізу тих рішень, що вже присутні на ринку програмних продуктів.

На наступних етапах було ознайомлення з літературою та поглиблення розуміння стосовно необхідних технологій для розробки програмного забезпечення.

Було зроблено попередній висновок, що досліджені нами аналоги не відповідають повною мірою заявленим вимогам користувачів. На підставі чого було зроблено висновок, що актуальним буде розробка нового застосунку з урахуванням існуючих недоліків.

Головними етапами стали «Проектування» та «Тестування та налагодження». У цих етапах дуже докладно описано усі технології які було використано під час розробки. Докладно пояснено як проектувалась система та як проводилось тестування.

Таким чином користувач отримує потужний інструмент розробки програмного забезпечення для автоматизованого проектування силових трансформаторів

СПИСОК ЛІТЕРАТУРИ

1. Автоматизация проектирования системы электроснабжения. Киев : Выща шк. Головное изд-во, 1988. 208 с.
2. Анцев, И.Б. Проектирование внутренних электрических сетей: учеб. Пособие. *Международный журнал экспериментального образования.* 2015. № 5-2. С. 217-218.
3. Бабенко В.В., Гольчевский Ю.В. Выбор языков программирования и средств проектирования для обучения специалистов по направлению «Прикладная информатика». *Прикладная информатика*. 2013. №4 (46). С. 43-45.
4. Елисеев, Д.С. Алгоритмы САПР для выбора проводов и кабелей. Волгоград: ВГАУ, 2012. 184 с.
5. Жалнин Р. В., Панюшкина Е. Н., Пескова Е. Е., Шаманаев П. А. Основы параллельного программирования с использованием технологий MPI и OpenMP. Саранск: СВМО, 2013. 78 с.
6. ‘C++ Primer’ by Stanley B. Lippman, Josée Lajoie, and Barbara E.
7. Мавлютов А. Р., Выдрин Д. Ф., Махнёва А. О. Самые востребованные языки программирования. *Academy*. 2017. №1 (16). С. 12-14.
8. Павлюков, В.А. Учебная САПР электрической части станций и подстанций Донецк: ДНТУ, 2016. 124 с.
9. Попов Г. В., Игнатьев Е. Б. Об определении индекса технического состояния силовых трансформаторов в процессе их эксплуатации. *Вестник Ивановского государственного энергетического университета.* 2014. № 4. С. 54-57.
10. Свиридов В. А., Бахарев Н. П. Обеспечение электродинамической стойкости силовых трансформаторов. *Молодой ученый.* 2017. № 32 (166). С. 20-25. URL: https://moluch.ru/archive/166/45370/ (дата обращения: 21.04.2020).

11. Стулов А. В., Трофимович И. А., Тихонов А. И. Разработка САПР силовых трансформаторов на основе автономных библиотек моделирования физических полей
и электрических цепей. *Пром-Инжиниринг: труды III Международной научно-технической конференции (Санкт-Петербург - Челябинск - Новочеркасск - Владивосток, 16-19 мая 2017 г.)*. Челябинск: Издательский центр ЮУрГУ, 2017. С. 78-84.

12. Тестування URL: https://www.browserstack.com/guide/software-testing-strategies-and-approaches

13. Тихомиров П. М. Расчет трансформаторов. Москва: ЛЕЛАНД, 2014.
528 с.

14. Язык программирования С++.Специальное издание / Под ред. Бьерн Страуструп - М., Бином 2011 - 45 с.

15. Hands-On High Performance Programming with Qt 5: Build cross-platform applications using concurrency, parallel programming, and memory management.

16. ООП. URL: https://stepik.org/lesson/341540/step/1 (Дата звернення:
16.05.2022)

17. Архітектура. URL: https://www.educative.io/blog/how-to-design-a-web-application-software-architecture-101#what (Дата звернення: 16.05.2022)

18. Архітектура. URL: https://www.outsystems.com/blog/posts/application-architecture/ (Дата звернення: 16.05.2022)

19. URL: https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/ruchnoe-testirovanie/chernyi-iashchik-protiv-belaia-korobka (Дата звернення: 16.05.2022)

20. URL: https://sites.google.com/site/testprogrammprodukt/home/testirovanie-belogo-i-cernogo-asika (Дата звернення: 16.05.2022)

21. URL: https://otus.ru/nest/post/605/ (Дата звернення: 16.05.2022)

22. URL: https://ravesli.com/urok-105-stek-i-kucha/ (Дата звернення: 16.05.2022)

23. URL: http://web.spt42.ru/index.php/chto-takoe-c-plus-plus (Дата звернення: 16.05.2022)

ЗАТВЕРДЖЕНО

44165850.01229-07 12 01 –ЛЗ

Автоматизоване проектування силових трансформаторів

Текст програми

44165850.01229-07 12 01-ЛЗ

Листів 28

2022

Анотація

Документ 44165850.01229-07 12 01 «Автоматизоване проектування силових трансформаторів» входить до складу програмної документації на програму, яка призначена для проектування силових трансформаторів та функціонує як складова частина навчально-проектної САПР електричних машин.

У даному документі описані функціональне призначення і структура програмного продукту, опис вхідних і вихідних даних, а також порядок установки програми. Описувана програма написана мовою С++ для IBM сумісних комп'ютерів. Обсяг пам'яті, займаний програмою, складає 1,5 Мбайт приблизно . Конфігурація комп'ютера стандартна. Програма функціонує в середовищі Windows 8/10.

ЗМІСТ

1 Короткий опис модулів програми

– Funtions.cpp – модуль призначений для розрахунку всіх функцій і запису їх у файл.

– Prototipes.h – модуль призначений для прототипування змінних

– Tran.cpp – модуль призначений для налагоджнгя логіки форм

– Tran.h – модуль призначений для налагодженя логіки елементів форм

2 Текст програми

2.1 *Модуль Function.cpp*

```
//---------------------------------------------------------------------------
#include"Prototypes.h"
#include<iostream>

//Функція отримання діаметру описаної окружності стержня магнітопроводу
//Вхідні дані
//S_n - товщина листа, константа зі значенням 0.35
// Вихідні дані
//d_c - діаметр олисаної окружності стержня магнітопроводу
int get_d_c(const double& S_n) {

	int d_c = (int)(55 * pow(S_n / 3, 1 / 4));

	//Робимо таке округлення, щоб число націло ділилося на 10
	while (d_c % 10 != 0) {
		d_c--;
	}

	return d_c;
}

//Функція отримання значення активного перетину сталі
//Вхідні дані
//K_z - коефіцієнт заповнення площі ступінчатої фігури сталью, константа зі
значенням 0.91
//K_kr - коефіцієнт заповнення площі кола площею ступінчатої фігури,
константа зі значенням 0.8 / 0.93
//d_c - діаметр олисаної окружності стержня магнітопроводу
//Вихідні дані
//Отримане значення активного перетину сталі округленого до тисячних
double get_S_a(const double& K_z, const double& K_kr, const int& d_c) {

	return round((3.14 * pow(d_c, 2) * K_kr * K_z * pow(10, -2) / 4) *
1000) / 1000;

}

//Функція отримання значення 1 фазної напруги обмотки
//Вхідні дані
//U_l1 - лінійна напруга обмотки НН
//K_cx - коефіцієнт схеми, при Д - 1.732, при У - 1
//Вихідні дані
//Отримане значення 1 фазної напруги обмотки округлене до тисячних
double get_U_1f(const int& U_l1, const double& K_cx) {

	return round((U_l1 / K_cx) * 1000) / 1000;

}

//Функція отримання значення кількості витків в обмотці НН
//Вхідні дані
//S_a - значення активного перетину сталі
```

```
//B_c - значення індукції
//U_1f - значення 1 фазної напруги обмотки
//f - частота сітки
//Вихідні дані
//Отримане значення кількості витків в обмотці НН округлене до цілого
значення
int get_W1(const double& S_a, const double& B_c, const int& f, const double&
U_1f) {

	return (int)round(U_1f / 4.44 * pow(10, -4) * S_a * f * B_c);

}

//Функція отримання значення ЕДС витку
//Вхідні дані
//U_1f - значення 1 фазної напруги обмотки
//W_1 - значення кількості витків в обмотці НН
//Вихідні дані
//Отримане значення ЕДС витку округлене до тисячних
double get_E_v(const double& U_1f, const int& W_1) {

	return round((U_1f / W_1) * 1000) / 1000;

}

//Функція отримання значення 2 фазної напруги обмотки
//Вхідні дані
//U_l2 - лінійна напруга обмотки ВН
//K_cx - коефіцієнт схеми, при Д - 1.732, при У - 1
//Вихідні дані
//Отримане значення 2 фазної напруги обмотки округлене до тисячних
double get_U_2f(const int& U_l2, const double& K_cx) {

	return round((U_l2 / K_cx) * 1000) / 1000;

}

//Функція отримання значення кількості витків в обмотці ВН
//Вхідні дані
//U_2f - значення 2 фазної напруги обмотки
//E_v - значення ЕДС
//Вихідні дані
//Отримане значення кількості витків в обмотці ВН округлене до цілого
значення
int get_W2(const double& U_2f, const double& E_v) {

	return (int)round(U_2f / E_v);

}

//Функція отримання максимального значення кількості витків в обмотці НН
//Вхідні дані
//W_2 - значення кількості витків в обмотці ВН
//n_ct - число ступенів регулювання
//c_ct - процент регулювання напруги
//Вихідні дані
//Отримане значення кількості витків в обмотці ВН округлене до цілого
значення
int get_W2_max(const int& W_2, const int& n_ct, const double& c_ct) {
```

```
	return (int)round(W_2 * (1 + 0.01 * n_ct * c_ct));

}

//Функція отримання мінімального значення кількості витків в обмотці НН
//Вхідні дані
//W_2 - значення кількості витків в обмотці ВН
//n_ct - число ступенів регулювання
//c_ct - процент регулювання напруги
//Вихідні дані
//Отримане значення кількості витків в обмотці ВН округлене до цілого
значення
int get_W2_min(const int& W_2, const int& n_ct, const double& c_ct) {

	return (int)round(W_2 * (1 - 0.01 * n_ct * c_ct));

}

//Функція отримання значення 1 фазного току обмотки
//Вхідні дані
//S_n - товщина листа, константа зі значенням 0.35
//U_1f - значення 1 фазної напруги обмотки
//Вихідні дані
//Отримане значення 1 фазного току обмотки округлене до тисячних
double get_I_1f(const double& S_n, const double& U_1f) {

	return round((S_n * pow(10, 3)) / (3 * U_1f) * 1000) / 1000;

}

//Функція отримання значення 2 фазного току обмотки
//Вхідні дані
//S_n - товщина листа, константа зі значенням 0.35
//U_2f - значення 1 фазної напруги обмотки
//Вихідні дані
//Отримане значення 2 фазного току обмотки округлене до тисячних
double get_I_2f(const double& S_n, const double& U_2f) {

	return round((S_n * pow(10, 3)) / (3 * U_2f) * 1000) / 1000;

}

//Функція отримання значення перетину обмотки НН
//Вхідні дані
//I_1f - значення 1 фазного току обмотки
//sigma_1 - принята перша плотність току, константа зі значенням 2
//Вихідні дані
//Отримане значення перетину обмотки НН округлене до тисячних
double get_S_1v(const double& I_1f, const double& sigma_1) {

	return round((I_1f / sigma_1) * 1000) / 1000;

}

//Функція отримання значення перетину обмотки ВН
//Вхідні дані
//I_2f - значення 2 фазного току обмотки
//sigma_2 - принята друга плотність току, константа зі значенням 3.5
```

```
//Вихідні дані
//Отримане значення перетину обмотки ВН округлене до тисячних
double get_S_2v(const double& I_2f, const double& sigma_2) {

	return round((I_2f / sigma_2) * 1000) / 1000;

}

//Функція отримання значення висоти обмоток
//Вхідні дані
//S_1v - значення перетину обмотки НН
//W_1 - значення кількості витків в обмотці НН
//b_iz1 - 1 товщина міжслоїовой ізоляції, константа зі значенням 0.35
//Вихідні дані
//Отримане значення висоти обмоток округлене до тисячних
double get_H1(const double& S_1v, const int& W_1, const double& b_iz1) {

	return round((1.05 * 1.1 * S_1v * W_1 / b_iz1) * 1000) / 1000;

}

//Функція отримамння значення 2 товщини ізоляціїної обмотки
//Вхідні дані
//W_2_max - максимальне значення кількості витків в обмотці НН
//H_1 - значення висоти обмоток
//Вихідні дані
//Отримане значення 2 товщини ізоляційної обмотки округлене до тисячних
double get_b_iz2(const int& W_2_max, const double& H_1) {

	return round((4 * 0.12 * (7 * W_2_max / (H_1 - 2)) * 1000)) / 1000;

}

//Функція отримання значення ширини обмотки ВН
//Вхідні дані
//S_2v - значення перетину обмотки ВН
//W_2_max - максимальне значення кількості витків в обмотці НН
//H_1 - значення висоти обмоток
//b_iz2 - значення 2 товщини ізоляціїної обмотки
//Вихідні дані
//Отримане значення ширини обмотки ВН округлене до тисячних
double get_a_2(const double& S_2v, const int& W_2_max, const double& H_1,
const double& b_iz2) {

	return round(((1.05 * 1.1 * S_2v * W_2_max / H_1) + b_iz2) * 1000) /
1000;

}

//Функція отримання значення середьного діматеру каналу розсіяння
//Вхідні дані
//d_c - діаметр описаної окружності стержня магнітопроводу
//a_01 - ізоляційна відстань між стержнем та обмоткою НН,константа зі
значенням 20
//a_1 - 0
//a_12 - ізоляційна відстань між обмотками НН та ВН,константа зі значенням 20
//Вихідні дані
//Отримане значення середьного діматеру каналу розсіяння округлене до
тисячних
```

```
double get_D_cp(const int& d_c, const int& a_01, const int& a_1, const int&
a_12) {

	return round((d_c + 2 * a_01 + 2 * a_1 + a_12) * 1000) / 1000;

}

//Функція отримання значення ширини приведенного каналу розсіювання
//Вхідні дані
//a_1 - 0
//a_2 - значення ширини обмотки ВН
//a_12 - ізоляційна відстань між обмотками НН та ВН,константа зі значенням 20
//Вихідні дані
//Отримане значення ширини приведенного каналу розсіювання округлене до
тисячних
double get_t(const int& a_1, const double& a_2, const int& a_12) {

	return round(((a_1 + a_2) / 3 + a_12) * 1000) / 1000;

}

//Функція отримання значення коефіцієнту Роговського
//Вхідні дані
//a_1 - 0
//a_2 - значення ширини обмотки ВН
//a_12 -  ізоляційна відстань між обмотками НН та ВН,константа зі значенням
20
//H_1 - значення висоти обмоток
//Вихідні дані
//Отримане значення коефіцієнту Роговського округлене до тисячних
double get_P_p(const int& a_1, const double& a_2, const int& a_12, const
double& H_1) {

	return round(((1 - (a_1 + a_2 + a_12)) / 3.14 * H_1) * 1000) / 1000;

}

//Функція отримання значення напруги короткого замикання
//Вхідні дані
//I_1f - значення 2 фазного току обмотки
//W_1 - значення кількості витків в обмотці НН
//f - частота сітки
//D_cp - значення середьного діматеру каналу розсіяння
//t - значення ширини приведенного каналу розсіювання
//P_p - значення коефіцієнту Роговського
//E_v - значення ЕДС витку
//H_1 - значення висоти обмоток
//Вихідні дані
//Отримане значення напруги короткого замикання округлене до тисячних
double get_U_k(const double& I_1f, const int& W_1, const int& f, const
double& D_cp, const double& t, const double& P_p, const double& E_v, const
double& H_1) {

	return -1 * round((I_1f * W_1 * f * D_cp * t * P_p * 1.02) / (40.5 *
pow(10, 3) * E_v * H_1) * 1000) / 1000;

}

//Функція отримання основних втрат короткого замикання
```

```
//Вхідні дані
//k_1 - коефіцієнт матеріалу, константа зі значенням 2.4
//sigma_1 - принята перша плотність току, константа зі значенням 2
//y - удільна плотність матеріалу
//d_c - діаметр олисаної окружності стержня магнітопроводу
//a_01 - ізоляційна відстань між стержнем та обмоткою НН,константа зі
значенням 20
//a_1 - 0
//W_1 - значення кількості витків в обмотці НН
//S_1v - значення перетину обмотки НН
//Вихідні дані
//Отримане значення основних втрат короткого замикання округлене до тисячних
double get_P_1(const double& k_1, const double& sigma_1, const double& y,
const int& d_c, const int& a_01, const int& a_1, const int& W_1, const
double& S_1v) {

	return round((k_1 * pow(sigma_1, 2) * 3 * 3.14 * y * (d_c + 2 * a_01 +
a_1) * W_1 * S_1v * pow(10, -6) * 1000)) / 1000;

}

//Функція отримання значення теплової обмотки НН
//Вхідні дані
//P_1 - значення основних втрат короткого замикання
//d_c - діаметр олисаної окружності стержня магнітопроводу
//a_01 - ізоляційна відстань між стержнем та обмоткою НН,константа зі
значенням 20
//a_1 - 0
//H_1 - значення висоти обмоток
//Вихідні дані
//Отримане значення теплової обмотки НН округлене до тисячних
double get_Q_1_HH(const double& P_1, const int& d_c, const int& a_01, const
int& a_1, const double& H_1) {

	return (0.5 * P_1 * pow(10, -3) * 1.3) / (3 * 3.14 * (d_c + 2 * a_01 +
a_1) * H_1 * pow(10, -3));

}

//Функція отримання значення перевищення температури обмотки
//Вхідні дані
//Q_1 - значення теплової обмотки НН
//a_v1 - внутрішній вертикальний охолоджуюючий канал, константа зі значенням
5
//a_n1 - зовнішній вертикальний охолоджуюючий канал, константа зі значенням
10
//t_m - перевищення температури верхніх слоїв масла,константа зі значенням 60
//Вихідні дані
//Отримане значення перевищення температури обмотки округлене до тисячних
double get_t_1_HH(const double& Q_1_HH, const int& a_v1, const int& a_n1,
const int& t_m) {

	return round((1.54 * pow(Q_1_HH, 6 / 10) * pow((a_v1 + a_n1), -35 /
100) * pow(t_m, -31 / 100) + 6) * 1000) / 1000;

}

//Функція отримання значення теплової обмотки для масла
//Вхідні дані
```

```
//k_2 - коефіцієнт матеріалу, константа зі значенням 21
//I_1f - значення 1 фазного току обмотки
//sigma_1 - принята перша плотність току, константа зі значенням 2
//W_1 - значення кількості витків в обмотці НН
//a_1 - 0
//Вихідні дані
//Отримане значення теплової обмотки для масла округлене до тисячних
double get_Q_1_oil(const int& k_2, const double& I_1f, const double& sigma_1,
const double& W_1, const int& a_1) {

	return round(1.5 * k_2 * I_1f * sigma_1 * W_1 / (2 * (a_1 + 10)) *
1000) / 1000;

}

//Функція отримання значенння перевищення температури над маслом для обмотки
катушки
//Вхідні дані
//Q_1_oil - значення теплової обмотки для масла
//a_1 - 0
//h_k1 - мінімальна висота горизонтального охолоджуючого каналу, константа зі
значенням 3
//a_v1 - внутрішній вертикальний охолоджуюючий канал, константа зі значенням
5
//a_n1 - зовнішній вертикальний охолоджуюючий канал, константа зі значенням
10
//t_m - перевищення температури верхніх слоїв масла,константа зі значенням 60
//Вихідні дані
//Отримане значення перевищення температури над маслом для обмотки катушки
округлене до тисячних
double get_t_1_oil(const double& Q_1_oil, const int& a_1, const int& h_k1,
const int& a_v1, const int& a_n1, const int& t_m) {

	return round((1.502 * pow(Q_1_oil, 6 / 10) * pow((a_1 + 0.01) / h_k1, 1
/ 4) * pow(a_v1, -1 / 4) * pow(a_n1, 21 / 100) * pow(t_m, -0.3)) * 1000) /
1000;

}

//Функція отримання значення висоти проводу
//Вхідні дані
//H_1 - значення висоти обмоток
//N - кількість слоїв, константа зі значенням 1
//W - значення кількості витків в обмотці
//Вихідні дані
//Отримане значення висоти проводу округлене до тисячних
double get_b_pr(const double& H_1, const int& N, const int& W) {

	return round((H_1 * N / W) * 1000) / 1000;

}

//Функція отримання значення ширини проводу
//Вхідні дані
//N - кількість слоїв, константа зі значенням 1
//a - значення ширини обмотки ВН
//b_k - ширина вертикального охолоджуючого каналу
//N_k - кількість вертикальних охолоджуючих каналів, константа зі значенням 1
//b_iz - значення товщини ізоляційної обмотки
```

```
//Вихідні дані
//Отримане значення ширини проводу округлене до тисячних
double get_a_pr(const int& N, const double& a, const double& b_k, const int&
N_k, const double& b_iz) {

	return round(((1 / N) * a - b_k * N_k - b_iz * (N - N_k - 1)) * 1000) /
1000;

}

//Функція отримання значення перетину проводу
//Вхідні дані
//a_pr - значення ширини проводу
//b_pr - значення висоти проводу
//Вихідні дані
//Отримане значення перетину проводу округлене до тисячних
double get_S(const double& a_pr, const double& b_pr) {

	return round((0.98 * a_pr * b_pr) * 1000) / 1000;

}

//Функція отримання значення перетину витку
//Вхідні дані
//x - 1
//S - значення перетину проводу
//Вихідні дані
//Отримане значення перетину витку округлене до тисячних
double get_S_v(const int& x, const double& S) {

	return round((S * x) * 1000) / 1000;

}

//Функція отримання значення плотності току
//Вхідні дані
//I_f - значення фазового току
//S_v - значення перетину витку
//Вихідні дані
//Отримане значення плотності току округлене до тисячних
double get_sigma(const double& I_f, const double& S_v) {

	return round((I_f / S_v) * 1000) / 1000;

}

//Функція отримання значення продольної зіставної поля розсіювання
//Вхідні дані
//P_p - коефіцієнт Роговського
//I_1f - значення 1 фазного току обмотки
//W_1 - значення кількості витків в обмотці НН
//H_1 - значення висоти обмоток
//Вихідні дані
//Отримане значення продольної зіставної поля розсіювання округлене до
тисячних
double get_B(const double& P_p, const double& I_1f, const int& W_1, const
double& H_1) {

	return round((1.78 * ((P_p * I_1f * W_1) / H_1)) * 1000) / 1000;
```

```
}

//Функція отримання значення додаткових втрат в обмотці
//Вхідні дані
//k_5 - коефіцієнт матеріалу, константа зі значенням 9.04
//sigma - значення плотності току
//a_pr - значення ширини проводу
//B - значення продольної зіставної поля розсіювання
//f - частота сітки
//Вихідні дані
//Отримане значення додаткових втрат в обмотці округлене до тисячних
double get_P_add(const double& k_5, const double& sigma, const double& a_pr,
const double& B, const int& f) {

	return (k_5 / 3) * pow(a_pr / pow(10, 4) / sigma, 2) * pow(B, 2) *
pow(f / 50, 2) * pow(10, -6);

}

//Функція отримання значення глубини проникнення
//Вхідні дані
//f - частота сітки
//a_50 - константа зі занчення 10.4
//Вихідні дані
//Отримане значення глубини проникнення округлене до тисячних
double get_a(const int& f, const double& a_50) {

	return  round((sqrt(50 / f * a_50) * 1000)) / 1000;

}

//Функція отримання значення коефіценту заповнення обмотки в осьовому
направленні
//Вхідні дані
//b_pr - значення висоти проводу
//H_1 - значення висоти обмоток
//Вихідні дані
//Отримане значення коефіцієнту заповнення обмотки в осьовому направленні
округлене до тисячних
double get_k_ob(const double& b_pr, const double& H_1) {

	return round((b_pr / H_1) * 1000) / 1000;

}

//Функція отримання середнього діаметру обмотки НН
//Вхідні дані
//d_c - діаметр олисаної окружності стержня магнітопроводу
//a_01 - ізоляційна відстань між стержнем та обмоткою НН,константа зі
значенням 20
//a_1 - 0
//Вихідні дані
//Отримане значення середнього діаметру обмотки НН округлене до тисячних
double get_D_cp1(const double& d_c, const double& a_01, const double& a_1) {

	return round((d_c + 2 * a_01 * a_1) * 1000) / 1000;

}
```

```
//Функція отримання значення середнього діаметру обмотки ВН
//Вхідні дані
//d_c - діаметр олисаної окружності стержня магнітопроводу
//a_01 - ізоляційна відстань між стержнем та обмоткою НН,константа зі
значенням 20
//a_1 - 0
//a_12 - ізоляційна відстань між обмотками НН та ВН,константа зі значенням 20
//a_2 - ширина обмотки ВН
//Вихідні дані
//Отримане значення середнього діаметру обмотки ВН округлене до тисячних
double get_D_cp2(const double& d_c, const double& a_01, const double& a_1,
const double& a_12, const double& a_2) {

	return round((d_c + 2 * (a_01 + a_1 + a_12) + a_2) * 1000) / 1000;

}

//Функція отримання довжини проводу обмотки
//Вхідні дані
//d_cp - середє арифметичне D_cp1 та D_cp2
//W - середнє арифметичне W_1 та W_2
//x - 1
//n - 1
//Вихідні дані
//отримане значення довжини проводу обмотки округлене до тисячних
double get_L(const double& d_cp, const int& W, const double& x, const double&
n) {

	return round((3.142 * d_cp * W * x * n * pow(10, -3)) * 1000) / 1000;

}

//Функція отримання активного спротиву обмотки
//Вхідні дані
//p - удільний спротив матеріалу обмотки, константа зі значенням 0.0214
//L - довжина проводу обмотки
//S_v - значення перетину витку
//n - 1
//x - 1
//Вихідні дані
//отримане значення активног оспротив обмотки округлене до тисячних
double get_r(const double& p, const double& L, const double& S_v, const
double& n, const double& x) {

	return round((p * L / (S_v * n * x)) * 1000) / 1000;

}

//Функція отримання втрат для обмотки НН
//Вхідні дані
//I_1f - значення 1 фазного току обмотки
//r - значення активного спротиву обмотки
//Вихідні дані
//Отримане значення втрат для обмотки НН
double get_P_1(const double& I_1f, const double& r) {

	return (3 * pow(I_1f, 2) * r);
```

```
}

//Функція отримання втрат для обмотки ВН
//Вхідні дані
//I_2f - значення 2 фазного току обмотки
//r - значення активного спротиву обмотки
//W_2 - значення кількості витків в обмотці ВН
//W_2_max - максимальне значення кількості витків в обмотці НН
//Вихідні дані
//Отримане значення втрат для обмотки ВН округлене до тисячних
double get_P_2(const double& I_2f, const double& r, const int& W_2, const
double& W_2_max) {

	return round((3 * pow(I_2f, 2) * r * (W_2 / W_2_max)) * 1000) / 1000;

}

//Функція отримання значення втрат від циркулюючих токів у результаті
небездоганності транспозиції
//Вхідні дані
//a_pr - значення ширини проводу
//b_iz - значення 2 товщини ізоляціїної обмотки
//k_ob - значення коефіценту заповнення обмотки в осьовому направленні
//a - значення глубини проникнення
//n - 1
//n_pr - кількість груп проводів в групових транспозиціях, константа зі
значенням 3
//P_0 - середнє арифметичне втрат p_1 та p_2
//Вихідні дані
//отримане значення втрат від циркулюючих токів у результаті небездоганності
транспозиції округлене до тисячних
double get_P_c(const double& a_pr, const double& b_iz, const double& k_ob,
const double& a, const int& n, const int& n_pr, const double& P_0) {

	//Працює тільки з частотою 50
	return round((0.02 * pow(a_pr / a * sqrt(k_ob), 4) * pow(a_pr + b_iz /
a_pr, 2) * (pow(n / n_pr, 2) - 1) * (pow(n / 2 * n_pr, 2) - 1) * P_0) * 1000)
/ 1000;

}

//Функція отримання критичної напруги за умовами радіальної стійкості
//Вхідні дані
//k8 - коефіціент, який залежить від матеріалу обмотки, константа зі
значенням 1.4
//k1 - коефіціент залежнйи від кількості прокладок, константа зі значенням
1.2
//D_cp - значення середьного діматеру каналу розсіяння
//k_2 - коефіціент залежний від числа рейок, константа зі значенням 0.3
//a_pr - значення ширини проводу
//b_pr - значення висоти проводу
//Вихідні дані
//Отримане значення критичної напруги за умовами радіальної стійкості
округлене до тисячних
double get_sigma_kr(const double& k8, const double& k1, const double& D_cp,
const double& k2, const double& a_pr, const double& b_pr) {

	return round((k8 * k1 * D_cp * (1 + k2 * a_pr) * (35 - b_pr) * pow(10,
3)) * 1000) / 1000;
```

```
}

//Функція отримання значення об`єму плотності теплового потоку
//Вхідні дані
//P_k - 1
//a - значення глубини проникнення
//D_cp - значення середьного діматеру каналу розсіяння
//H -  значення висоти обмоток
//Вихідні дані
//Отримане значення об`єму плотності теплового потоку округлене до тисячних
double get_q_v(const double& P_k, const double& a, const double& D_cp, const
double& H) {

	return round(((P_k * pow(10, 9)) / (3.14 * a * D_cp * H)) * 1000) /
1000;

}

//Функція отримання коефіцієнту закриття поверхні обмотки рейками
//Вхідні дані
//D_cp - значення середьного діматеру каналу розсіяння
//b_pr - значення висоти проводу
//n_pr - кількість груп проводів в групових транспозиціях, константа зі
значенням 3
//Вихідні дані
//Отримане значення коефіцієнту закриття поверхні обмотки рейками округлене
до тисячних
double get_k_z(const double& D_cp, const double& b_pr, const double& n_pr) {

	return round(((3.14 * D_cp) / (3.14 * D_cp - 0.5 * b_pr * n_pr)) *
1000) / 1000;

}

//Функція отримання значення поверхневої плотності теплового потоку
//Вхідні дані
//q_v - значення об`єму плотності теплового потоку
//a - значення глубини проникнення
//k_z -значення коефіцієнту закриття поверхні обмотки рейками
//Вихідні дані
//Отримане значення поверхневої плотності теплового потоку округлене до
тисячних
double get_q(const double& q_v, const double& a, const double& k_z) {

	return round((0.5 * q_v * a * k_z * pow(10, -3)) * 1000) / 1000;

}

//Функція запису даних для запису до бази даних до структури
//Вхідні дані
//d_c - значення діаметру описаної окружності стержня магнітопроводу
//W_1 - значення кількості витків в обмотці НН
//W_2 - значення кількості витків в обмотці ВН
//W_2_max - значення максимальної кількості витків в обмотці ВН
//W_2_min - значення мінімальної кількості витків в обмотці ВН
//S_a - значення активного перетину сталі
//E_v - значення ЕДС витку
//S_1v - значення перетину витків обмотки НН
```

```
//S_2v - значення перетину витків обмотки ВН
//H_1 - значення висоти обмоток
//a_2 - значення ширини обмотки ВН
//U_k - значення напруги короткого замикання
//t_1_HH - значення перевишення температури обмотки НН
//t_1_oil - значення перевишення температури над маслом для обмотки катушки
//P_add - значення додаткових втрат в обмотці
//P_c - значення втрат от циркулюючих токів у результаті небездоганності
транспозиції
//sigma_kr - значення критичної напруги по умовам радіальної стійкості
//q - значення плотності теплового потоку
//Вихідні дані
//Заповнена структура з даними для запису до бази данних
values_to_file get_values_to_struct(const int&d_c, const int&W_1, const
int&W_2, const int&W_2_max, const int&W_2_min, const double&S_a, const
double&E_v, const double&S_1v, const double&S_2v, const double&H_1, const
double&a_2, const double&U_k, const double&t_1_HH, const double&t_1_oil,
const double&P_add, const double&P_c, const double&sigma_kr,const double&q) {
      values_to_file struct_save;
      struct_save.d_c = d_c;
      struct_save.W_1 = W_1;
      struct_save.W_2 = W_2;
      struct_save.W_2_max = W_2_max;
      struct_save.W_2_min = W_2_min;
      struct_save.S_a = S_a;
      struct_save.E_v = E_v;
      struct_save.S_1v = S_2v;
      struct_save.H_1 = H_1;
      struct_save.a_2 = a_2;
      struct_save.U_k = U_k;
      struct_save.t_1_HH = t_1_HH;
      struct_save.t_1_oil = t_1_oil;
      struct_save.P_add = P_add;
      struct_save.P_c = P_c;
      struct_save.sigma_kr = sigma_kr;
      struct_save.q = q;

      return struct_save;
}

//Функція запису даних до бази даних у вигляді текстового файлу
//Вхідні дані
//struct_to_write - структура з даними, які будуть записані до файлу
//DATA_BASE - файл для запису даних
//Вихідні дані(скоріше вихідна дія)
//Запис усіх даних зі структури до файлу
void write_to_file(const values_to_file&struct_to_write,FILE*&DATA_BASE) {
      fputs("Значення діаметру описаної окружності стержня магнітопроводу :
", DATA_BASE);
      fprintf(DATA_BASE, "%d", struct_to_write.d_c);
      fputs("\n", DATA_BASE);

      fputs("Значення активного перетину сталі : ", DATA_BASE);
      fprintf(DATA_BASE, "%le", struct_to_write.S_a);
      fputs("\n", DATA_BASE);

      fputs("Значення ЕДС витку : ", DATA_BASE);
      fprintf(DATA_BASE, "%le", struct_to_write.E_v);
      fputs("\n", DATA_BASE);
```

```
      fputs("Значення кількості витків в обмотці НН : ", DATA_BASE);
      fprintf(DATA_BASE, "%d", struct_to_write.W_1);
      fputs("\n", DATA_BASE);

      fputs("Значення кількості витків в обмотці ВН : ", DATA_BASE);
      fprintf(DATA_BASE, "%d", struct_to_write.W_2);
      fputs("\n", DATA_BASE);

      fputs("Значення максимальної кількості витків в обмотці ВН : ",
DATA_BASE);
      fprintf(DATA_BASE, "%d", struct_to_write.W_2_max);
      fputs("\n", DATA_BASE);

      fputs("Значення мінімальної кількості витків в обмотці ВН : ",
DATA_BASE);
      fprintf(DATA_BASE, "%d", struct_to_write.W_2_min);
      fputs("\n", DATA_BASE);

      fputs("Значення перетину витків обмотки НН : ", DATA_BASE);
      fprintf(DATA_BASE, "%le", struct_to_write.S_1v);
      fputs("\n", DATA_BASE);

      fputs("Значення перетину витків обмотки ВН : ", DATA_BASE);
      fprintf(DATA_BASE, "%le", struct_to_write.S_2v);
      fputs("\n", DATA_BASE);

      fputs("Значення висоти обмоток : ", DATA_BASE);
      fprintf(DATA_BASE, "%le", struct_to_write.H_1);
      fputs("\n", DATA_BASE);

      fputs("Значення ширини обмотки ВН : ", DATA_BASE);
      fprintf(DATA_BASE, "%le", struct_to_write.a_2);
      fputs("\n", DATA_BASE);

      fputs("Значення напруги короткого замикання : ", DATA_BASE);
      fprintf(DATA_BASE, "%le", struct_to_write.U_k);
      fputs("\n", DATA_BASE);

      fputs("Значення перевишення температури обмотки НН : ", DATA_BASE);
      fprintf(DATA_BASE, "%le", struct_to_write.t_1_HH);
      fputs("\n", DATA_BASE);

      fputs("Значення перевишення температури над маслом для обмотки катушки
: ", DATA_BASE);
      fprintf(DATA_BASE, "%le", struct_to_write.t_1_oil);
      fputs("\n", DATA_BASE);

      fputs("Значення додаткових втрат в обмотці : ", DATA_BASE);
      fprintf(DATA_BASE, "%le", struct_to_write.P_add);
      fputs("\n", DATA_BASE);

      fputs("Значення втрат от циркулюючих токів у результаті небездоганності
транспозиції : ", DATA_BASE);
      fprintf(DATA_BASE, "%le", struct_to_write.P_c);
      fputs("\n", DATA_BASE);

      fputs("Значення критичної напруги по умовам радіальної стійкості : ",
DATA_BASE);
```

```
	fprintf(DATA_BASE, "%le", struct_to_write.sigma_kr);
	fputs("\n", DATA_BASE);

	fputs("Значення плотності теплового потоку : ", DATA_BASE);
	fprintf(DATA_BASE, "%le", struct_to_write.q);
	fputs("\n", DATA_BASE);

}
```

2.2 *Модуль Prototipes.h*

```
//---------------------------------------------------------------------------
# #pragma once
```

```
//Підключення макросу для використовування старих функцій
#define _CRT_SECURE_NO_WARNINGS

//Підключення бібліотеки для роботи з файлами
#include<fstream>

//Структура з даними, які мають бути записані до файлу
struct values_to_file {
	int d_c;
	int W_1;
	int W_2;
	int W_2_max;
	int W_2_min;
	double S_a;
	double E_v;
	double S_1v;
	double S_2v;
	double H_1;
	double a_2;
	double U_k;
	double t_1_HH;
	double t_1_oil;
	double P_add;
	double P_c;
	double sigma_kr;
	double q;
};

//Прототип функції отримання діаметру описаної окружності стержня магнітопроводу
int get_d_c(const double&);

//Прототип функції отримання значення активного перетину сталі
double get_S_a(const double&, const double&, const int&);

//Прототип функції отримання значення 1 фазної напруги обмотки
double get_U_1f(const int&, const double&);

//Прототип функції отримання значення кількості витків в обмотці НН
int get_W1(const double&, const double&, const int&, const double&);

//Прототип функції отримання значення ЕДС витку
double get_E_v(const double&, const int&);

//Прототип функції отримання значення 2 фазної напруги обмотки
double get_U_2f(const int&, const double&);

//Прототип функції отримання значення кількості витків в обмотці ВН
int get_W2(const double&, const double&);

//Прототип функції отримання значення максимальної кількості витків в обмотці ВН
int get_W2_max(const int&, const int&, const double&);

//Прототип функції отримання значення мінімальної кількості витків в обмотці ВН
int get_W2_min(const int&, const int&, const double&);

//Прототип функції отримання значення 1 фазного току обмотки
double get_I_1f(const double&, const double&);
```

```
//Прототип функції отримання значення 2 фазного току обмотки
double get_I_2f(const double&, const double&);

//Прототип функції отримання значення перетину обмотки НН
double get_S_1v(const double&, const double&);

//Прототип функції отримання значення перетину обмотки ВН
double get_S_2v(const double&, const double&);

//Прототип функцій отримання значення висоти обмоток
double get_H1(const double&, const int&, const double&);

//Прототип функції отримамння значення 2 товщини ізоляціїної обмотки
double get_b_iz2(const int&, const double&);

//Прототип функції отримання ширини обмотки ВН
double get_a_2(const double&, const int&, const double&, const double&);

//Прототип функції отримання значення середьного діматеру каналу розсіяння
double get_D_cp(const int&, const int&, const int&, const int&);

//Прототип функції отримання значення ширини приведенного каналу розсіювання
double get_t(const int&, const double&, const int&);

//Прототип функції отримання значення коефіцієнту Роговського
double get_P_p(const int&, const double&, const int&, const double&);

//Прототип функції значення напруги короткого замикання
double get_U_k(const double& I_1f, const int& W_1, const int& f, const double& D_cp, const double& t, const double&
P_p, const double& E_v, const double& H_1);

//Прототип функції отримання основних втрат короткого замикання
double get_P_1(const double&, const double&, const double&, const int&, const int&, const int&, const int&, const
double&);

//Прототип функції отримання значення теплової обмотки НН
double get_Q_1_HH(const double&, const int&, const int&, const int&, const double&);

//Прототип функції отримання значення перевищення температури обмотки
double get_t_1_HH(const double&, const int&, const int&, const int&);

//Прототип отримання значення теплової обмотки для масла
double get_Q_1_oil(const int&, const double&, const double&, const double&, const int&);

//Прототип функції отримання значеннння перевищення температури над маслом для обмотки катушки
double get_t_1_oil(const double&, const int&, const int&, const int&, const int&, const int&);

//Прототип функції отримання значення висоти проводу
double get_b_pr(const double&, const int&, const int&);

//Прототип функції отримання значення ширини проводу
double get_a_pr(const int&, const double&, const double&, const int&, const double&);

//Прототип функції отримання значення перетину проводу
double get_S(const double&, const double&);

//Прототип функції отримання значення перетину витку
```

```
double get_S_v(const int&, const double&);

//Прототип функції отримання значення плотності току
double get_sigma(const double&, const double&);

//Прототип функції отримання значення продольної зіставної поля розсіювання
double get_B(const double&, const double&, const int&, const double&);

//Прототип функції отримання значення додаткових втрат в обмотці
double get_P_add(const double&, const double&, const double&, const double&, const int&);

//Прототип функції отримання значення глубини проникнення
double get_a(const int&, const double&);

//Прототип функції заповнення обмотки в осьовому напрямку
double get_k_ob(const double&, const double&);

//Прототип функції отримання середнього діаметру обмотки НН
double get_D_cp1(const double&, const double&, const double&);

//Прототип функції отримання значення середнього діаметру обмотки ВН
double get_D_cp2(const double&, const double&, const double&, const double&, const double&);

//Прототип функції отримання довжини проводу обмотки
double get_L(const double&, const int&, const double&, const double&);

//Прототип функції отримання активного спротиву обмотки
double get_r(const double&, const double&, const double&, const double&, const double&);

//Прототип функції отримання втрат для обмотки НН
double get_P_1(const double&, const double&);

//Прототип функції отримання втрат для обмотки ВН
double get_P_2(const double&, const double&, const int&, const double&);

//Прототип функції отримання значення втрат від циркулюючих токів у результаті небездоганності транспозиції
double get_P_c(const double&, const double&, const double&, const double&, const int&, const int&, const double&);

//Прототип функції отримання критичної напруги за умовами радіальної стійкості
double get_sigma_kr(const double&, const double&, const double&, const double&, const double&, const double&);

//Прототип функції отримання значення об`єму плотності теплового потоку
double get_q_v(const double&, const double&, const double&, const double&);

//Прототип функції отримання коефіцієнту закриття поверхні обмотки рейками
double get_k_z(const double&, const double&, const double&);

//Прототип функції отримання значення поверхневої плотності теплового потоку
double get_q(const double&, const double&, const double&);

//Прототип функції запису даних для запису до бази даних до структури
values_to_file get_values_to_struct(const int&, const int&, const int&, const int&, const int&, const double&, const
double&, const double&, const double&, const double&, const double&, const double&, const double&, const double&,
const double&, const double&, const double&, const double&);

//Прототип функції запису даних до бази даних
void write_to_file(const values_to_file&, FILE*&);
```

2.3 *Модуль Tran.cpp*

```
#include "Tran.h"

using namespace System;
using namespace System::Windows::Forms;
[STAThreadAttribute]
int main(array<String^>^ args) {
        Application::EnableVisualStyles();
```

```
	Application::SetCompatibleTextRenderingDefault(false);
	practicetran::Tran form;
	Application::Run(% form);
	return 0;
}
```

2.4 *Модуль Tran.h*

```
#pragma once

#include"Prototypes.h"

namespace practicetran {

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

	/// <summary>
	/// Сводка для Tran
	/// </summary>
	public ref class Tran : public System::Windows::Forms::Form
	{
	public:
		Tran(void)
		{
			InitializeComponent();
			//
			//TODO: добавьте код конструктора
			//
		}

	protected:
		/// <summary>
		/// Освободить все используемые ресурсы.
		/// </summary>
		~Tran()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::Label^ label_mes_tran_type;
	private: System::Windows::Forms::Label^ label_tran_type;
	private: System::Windows::Forms::Label^ label_mes_tran_power;
	private: System::Windows::Forms::Label^ label_mes_tran_frequency;
	private: System::Windows::Forms::Label^ label_mes_tran_voltage_HH;

	private: System::Windows::Forms::Label^ label_mes_tran_voltage_BH;
	private: System::Windows::Forms::Label^ label_mes_tran_plan;
	private: System::Windows::Forms::Label^ label_mew_tran_stage;
	private: System::Windows::Forms::Label^ label_mew_tran_percent;
```

```
protected:

protected:
```

```
private: System::Windows::Forms::Button^ button_calculate;
private: System::Windows::Forms::Label^ label_mew_induction;
private: System::Windows::Forms::ComboBox^ comboBox_tran_power;

private: System::Windows::Forms::ComboBox^ comboBox_tran_voltage_HH;
private: System::Windows::Forms::ComboBox^ comboBox_tran_voltage_BH;
private: System::Windows::Forms::ComboBox^ comboBox_tran_plan;
private: System::Windows::Forms::ComboBox^ comboBox_tran_stage;
private: System::Windows::Forms::ComboBox^ comboBox_tran_percent;
private: System::Windows::Forms::ComboBox^ comboBox_tran_induction;
private: System::Windows::Forms::Label^ label_mes_mag_cir_type;
private: System::Windows::Forms::Label^ label_mag_cir_type;
private: System::Windows::Forms::Label^ label_mes_metal_type;
private: System::Windows::Forms::Label^ label_metal_type;
private: System::Windows::Forms::Label^ label_frequency;
```

```
private:
      /// <summary>
      /// Обязательная переменная конструктора.
```

```
            /// </summary>
            System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Требуемый метод для поддержки конструктора — не изменяйте
            /// содержимое этого метода с помощью редактора кода.
            /// </summary>
            void InitializeComponent(void)
            {
                  this->label_mes_tran_type = (gcnew
System::Windows::Forms::Label());
                  this->label_tran_type = (gcnew
System::Windows::Forms::Label());
                  this->label_mes_tran_power = (gcnew
System::Windows::Forms::Label());
                  this->label_mes_tran_frequency = (gcnew
System::Windows::Forms::Label());
                  this->label_mes_tran_voltage_HH = (gcnew
System::Windows::Forms::Label());
                  this->label_mes_tran_voltage_BH = (gcnew
System::Windows::Forms::Label());
                  this->label_mes_tran_plan = (gcnew
System::Windows::Forms::Label());
                  this->label_mew_tran_stage = (gcnew
System::Windows::Forms::Label());
                  this->label_mew_tran_percent = (gcnew
System::Windows::Forms::Label());
                  this->button_calculate = (gcnew
System::Windows::Forms::Button());
                  this->label_mew_induction = (gcnew
System::Windows::Forms::Label());
                  this->comboBox_tran_power = (gcnew
System::Windows::Forms::ComboBox());
                  this->comboBox_tran_voltage_HH = (gcnew
System::Windows::Forms::ComboBox());
                  this->comboBox_tran_voltage_BH = (gcnew
System::Windows::Forms::ComboBox());
                  this->comboBox_tran_plan = (gcnew
System::Windows::Forms::ComboBox());
                  this->comboBox_tran_stage = (gcnew
System::Windows::Forms::ComboBox());
                  this->comboBox_tran_percent = (gcnew
System::Windows::Forms::ComboBox());
                  this->comboBox_tran_induction = (gcnew
System::Windows::Forms::ComboBox());
                  this->label_mes_mag_cir_type = (gcnew
System::Windows::Forms::Label());
                  this->label_mag_cir_type = (gcnew
System::Windows::Forms::Label());
                  this->label_mes_metal_type = (gcnew
System::Windows::Forms::Label());
                  this->label_metal_type = (gcnew
System::Windows::Forms::Label());
                  this->label_frequency = (gcnew
System::Windows::Forms::Label());
                  this->SuspendLayout();
                  //
                  // label_mes_tran_type
```

```
                //
                this->label_mes_tran_type->AutoSize = true;
                this->label_mes_tran_type->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
                this->label_mes_tran_type->Location =
System::Drawing::Point(12, 9);
                this->label_mes_tran_type->Name = L"label_mes_tran_type";
                this->label_mes_tran_type->Size =
System::Drawing::Size(199, 20);
                this->label_mes_tran_type->TabIndex = 0;
                this->label_mes_tran_type->Text = L"Тип трансформатору :";
                //
                // label_tran_type
                //
                this->label_tran_type->AutoSize = true;
                this->label_tran_type->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.8F));
                this->label_tran_type->Location =
System::Drawing::Point(217, 9);
                this->label_tran_type->Name = L"label_tran_type";
                this->label_tran_type->Size = System::Drawing::Size(108,
20);
                this->label_tran_type->TabIndex = 1;
                this->label_tran_type->Text = L"ТМ - 4000/35";
                //
                // label_mes_tran_power
                //
                this->label_mes_tran_power->AutoSize = true;
                this->label_mes_tran_power->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
                this->label_mes_tran_power->Location =
System::Drawing::Point(12, 38);
                this->label_mes_tran_power->Name = L"label_mes_tran_power";
                this->label_mes_tran_power->Size =
System::Drawing::Size(343, 20);
                this->label_mes_tran_power->TabIndex = 2;
                this->label_mes_tran_power->Text = L"Виберіть потужність
трансформатору :";
                //
                // label_mes_tran_frequency
                //
                this->label_mes_tran_frequency->AutoSize = true;
                this->label_mes_tran_frequency->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
                this->label_mes_tran_frequency->Location =
System::Drawing::Point(12, 69);
                this->label_mes_tran_frequency->Name =
L"label_mes_tran_frequency";
                this->label_mes_tran_frequency->Size =
System::Drawing::Size(214, 20);
                this->label_mes_tran_frequency->TabIndex = 3;
                this->label_mes_tran_frequency->Text = L"Виберіть частоту
сітки :";
                //
                // label_mes_tran_voltage_HH
                //
                this->label_mes_tran_voltage_HH->AutoSize = true;
                this->label_mes_tran_voltage_HH->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
```

```
                this->label_mes_tran_voltage_HH->Location =
System::Drawing::Point(12, 100);
                this->label_mes_tran_voltage_HH->Name =
L"label_mes_tran_voltage_HH";
                this->label_mes_tran_voltage_HH->Size =
System::Drawing::Size(332, 20);
                this->label_mes_tran_voltage_HH->TabIndex = 4;
                this->label_mes_tran_voltage_HH->Text = L"Виберіть лінійну
напругу обмотки НН :";
                //
                // label_mes_tran_voltage_BH
                //
                this->label_mes_tran_voltage_BH->AutoSize = true;
                this->label_mes_tran_voltage_BH->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
                this->label_mes_tran_voltage_BH->Location =
System::Drawing::Point(12, 130);
                this->label_mes_tran_voltage_BH->Name =
L"label_mes_tran_voltage_BH";
                this->label_mes_tran_voltage_BH->Size =
System::Drawing::Size(331, 20);
                this->label_mes_tran_voltage_BH->TabIndex = 5;
                this->label_mes_tran_voltage_BH->Text = L"Виберіть лінійну
напругу обмотки ВН :";
                //
                // label_mes_tran_plan
                //
                this->label_mes_tran_plan->AutoSize = true;
                this->label_mes_tran_plan->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
                this->label_mes_tran_plan->Location =
System::Drawing::Point(13, 161);
                this->label_mes_tran_plan->Name = L"label_mes_tran_plan";
                this->label_mes_tran_plan->Size =
System::Drawing::Size(314, 20);
                this->label_mes_tran_plan->TabIndex = 6;
                this->label_mes_tran_plan->Text = L"Виберіть схему
з`єднання обмотки :";
                //
                // label_mew_tran_stage
                //
                this->label_mew_tran_stage->AutoSize = true;
                this->label_mew_tran_stage->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
                this->label_mew_tran_stage->Location =
System::Drawing::Point(12, 190);
                this->label_mew_tran_stage->Name = L"label_mew_tran_stage";
                this->label_mew_tran_stage->Size =
System::Drawing::Size(416, 20);
                this->label_mew_tran_stage->TabIndex = 7;
                this->label_mew_tran_stage->Text = L"Виберіть кількість
ступеней регулювання в ВН :";
                //
                // label_mew_tran_percent
                //
                this->label_mew_tran_percent->AutoSize = true;
                this->label_mew_tran_percent->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
```

```
                this->label_mew_tran_percent->Location =
System::Drawing::Point(13, 219);
                this->label_mew_tran_percent->Name =
L"label_mew_tran_percent";
                this->label_mew_tran_percent->Size =
System::Drawing::Size(366, 20);
                this->label_mew_tran_percent->TabIndex = 8;
                this->label_mew_tran_percent->Text = L"Виберіть процент
регулювання ступеней :";
                //
                // button_calculate
                //
                this->button_calculate->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.8F));
                this->button_calculate->Location =
System::Drawing::Point(15, 370);
                this->button_calculate->Name = L"button_calculate";
                this->button_calculate->Size = System::Drawing::Size(631,
28);
                this->button_calculate->TabIndex = 16;
                this->button_calculate->Text = L"Зробити обчислення ";
                this->button_calculate->UseVisualStyleBackColor = true;
                this->button_calculate->Click += gcnew
System::EventHandler(this, &Tran::button1_Click);
                //
                // label_mew_induction
                //
                this->label_mew_induction->AutoSize = true;
                this->label_mew_induction->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
                this->label_mew_induction->Location =
System::Drawing::Point(13, 249);
                this->label_mew_induction->Name = L"label_mew_induction";
                this->label_mew_induction->Size =
System::Drawing::Size(170, 20);
                this->label_mew_induction->TabIndex = 17;
                this->label_mew_induction->Text = L"Виберіть індукцію :";
                //
                // comboBox_tran_power
                //
                this->comboBox_tran_power->FormattingEnabled = true;
                this->comboBox_tran_power->Items->AddRange(gcnew
cli::array< System::Object^  >(6) {
                      L"4000", L"4200", L"4400", L"4600", L"4800",
                            L"5000"
                });
                this->comboBox_tran_power->Location =
System::Drawing::Point(441, 37);
                this->comboBox_tran_power->Name = L"comboBox_tran_power";
                this->comboBox_tran_power->Size =
System::Drawing::Size(205, 24);
                this->comboBox_tran_power->TabIndex = 19;
                //
                // comboBox_tran_voltage_HH
                //
                this->comboBox_tran_voltage_HH->FormattingEnabled = true;
                this->comboBox_tran_voltage_HH->Items->AddRange(gcnew
cli::array< System::Object^  >(5) {
                      L"6300", L"6500", L"6700", L"6900",
```

```
                        L"7100"
                });
                this->comboBox_tran_voltage_HH->Location =
System::Drawing::Point(441, 97);
                this->comboBox_tran_voltage_HH->Name =
L"comboBox_tran_voltage_HH";
                this->comboBox_tran_voltage_HH->Size =
System::Drawing::Size(205, 24);
                this->comboBox_tran_voltage_HH->TabIndex = 21;
                //
                // comboBox_tran_voltage_BH
                //
                this->comboBox_tran_voltage_BH->FormattingEnabled = true;
                this->comboBox_tran_voltage_BH->Items->AddRange(gcnew
cli::array< System::Object^  >(5) {
                        L"3500", L"4000", L"4500", L"5000",
                                L"5500"
                });
                this->comboBox_tran_voltage_BH->Location =
System::Drawing::Point(442, 127);
                this->comboBox_tran_voltage_BH->Name =
L"comboBox_tran_voltage_BH";
                this->comboBox_tran_voltage_BH->Size =
System::Drawing::Size(205, 24);
                this->comboBox_tran_voltage_BH->TabIndex = 22;
                //
                // comboBox_tran_plan
                //
                this->comboBox_tran_plan->FormattingEnabled = true;
                this->comboBox_tran_plan->Items->AddRange(gcnew cli::array<
System::Object^  >(2) { L"Д", L"У" });
                this->comboBox_tran_plan->Location =
System::Drawing::Point(441, 157);
                this->comboBox_tran_plan->Name = L"comboBox_tran_plan";
                this->comboBox_tran_plan->Size = System::Drawing::Size(205,
24);
                this->comboBox_tran_plan->TabIndex = 23;
                //
                // comboBox_tran_stage
                //
                this->comboBox_tran_stage->FormattingEnabled = true;
                this->comboBox_tran_stage->Items->AddRange(gcnew
cli::array< System::Object^  >(4) { L"2", L"3", L"4", L"5" });
                this->comboBox_tran_stage->Location =
System::Drawing::Point(441, 187);
                this->comboBox_tran_stage->Name = L"comboBox_tran_stage";
                this->comboBox_tran_stage->Size =
System::Drawing::Size(205, 24);
                this->comboBox_tran_stage->TabIndex = 24;
                //
                // comboBox_tran_percent
                //
                this->comboBox_tran_percent->FormattingEnabled = true;
                this->comboBox_tran_percent->Items->AddRange(gcnew
cli::array< System::Object^  >(6) {
                        L"2.5", L"3.0", L"3.5", L"4.0", L"4.5",
                                L"5.0"
                });
```

```
                this->comboBox_tran_percent->Location =
System::Drawing::Point(442, 218);
                this->comboBox_tran_percent->Name =
L"comboBox_tran_percent";
                this->comboBox_tran_percent->Size =
System::Drawing::Size(205, 24);
                this->comboBox_tran_percent->TabIndex = 25;
                //
                // comboBox_tran_induction
                //
                this->comboBox_tran_induction->FormattingEnabled = true;
                this->comboBox_tran_induction->Items->AddRange(gcnew
cli::array< System::Object^  >(6) {
                      L"1.40", L"1.42", L"1.44", L"1.46",
                            L"1.48", L"1.50"
                });
                this->comboBox_tran_induction->Location =
System::Drawing::Point(442, 248);
                this->comboBox_tran_induction->Name =
L"comboBox_tran_induction";
                this->comboBox_tran_induction->Size =
System::Drawing::Size(205, 24);
                this->comboBox_tran_induction->TabIndex = 26;
                //
                // label_mes_mag_cir_type
                //
                this->label_mes_mag_cir_type->AutoSize = true;
                this->label_mes_mag_cir_type->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
                this->label_mes_mag_cir_type->Location =
System::Drawing::Point(12, 279);
                this->label_mes_mag_cir_type->Name =
L"label_mes_mag_cir_type";
                this->label_mes_mag_cir_type->Size =
System::Drawing::Size(188, 20);
                this->label_mes_mag_cir_type->TabIndex = 27;
                this->label_mes_mag_cir_type->Text = L"Тип магнітопроводу
:";
                //
                // label_mag_cir_type
                //
                this->label_mag_cir_type->AutoSize = true;
                this->label_mag_cir_type->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
                this->label_mag_cir_type->Location =
System::Drawing::Point(438, 279);
                this->label_mag_cir_type->Name = L"label_mag_cir_type";
                this->label_mag_cir_type->Size = System::Drawing::Size(110,
20);
                this->label_mag_cir_type->TabIndex = 28;
                this->label_mag_cir_type->Text = L"стержневий";
                //
                // label_mes_metal_type
                //
                this->label_mes_metal_type->AutoSize = true;
                this->label_mes_metal_type->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
                this->label_mes_metal_type->Location =
System::Drawing::Point(13, 308);
```

```
                this->label_mes_metal_type->Name = L"label_mes_metal_type";
                this->label_mes_metal_type->Size =
System::Drawing::Size(119, 20);
                this->label_mes_metal_type->TabIndex = 29;
                this->label_mes_metal_type->Text = L"Тип металу  :";
                this->label_mes_metal_type->Click += gcnew
System::EventHandler(this, &Tran::label1_Click);
                //
                // label_metal_type
                //
                this->label_metal_type->AutoSize = true;
                this->label_metal_type->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
                this->label_metal_type->Location =
System::Drawing::Point(438, 308);
                this->label_metal_type->Name = L"label_metal_type";
                this->label_metal_type->Size = System::Drawing::Size(47,
20);
                this->label_metal_type->TabIndex = 30;
                this->label_metal_type->Text = L"мідь";
                //
                // label_frequency
                //
                this->label_frequency->AutoSize = true;
                this->label_frequency->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.9F));
                this->label_frequency->Location =
System::Drawing::Point(438, 69);
                this->label_frequency->Name = L"label_frequency";
                this->label_frequency->Size = System::Drawing::Size(27,
20);
                this->label_frequency->TabIndex = 31;
                this->label_frequency->Text = L"50";
                //
                // Tran
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(8, 16);
                this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
                this->ClientSize = System::Drawing::Size(658, 407);
                this->Controls->Add(this->label_frequency);
                this->Controls->Add(this->label_metal_type);
                this->Controls->Add(this->label_mes_metal_type);
                this->Controls->Add(this->label_mag_cir_type);
                this->Controls->Add(this->label_mes_mag_cir_type);
                this->Controls->Add(this->comboBox_tran_induction);
                this->Controls->Add(this->comboBox_tran_percent);
                this->Controls->Add(this->comboBox_tran_stage);
                this->Controls->Add(this->comboBox_tran_plan);
                this->Controls->Add(this->comboBox_tran_voltage_BH);
                this->Controls->Add(this->comboBox_tran_voltage_HH);
                this->Controls->Add(this->comboBox_tran_power);
                this->Controls->Add(this->label_mew_induction);
                this->Controls->Add(this->button_calculate);
                this->Controls->Add(this->label_mew_tran_percent);
                this->Controls->Add(this->label_mew_tran_stage);
                this->Controls->Add(this->label_mes_tran_plan);
                this->Controls->Add(this->label_mes_tran_voltage_BH);
                this->Controls->Add(this->label_mes_tran_voltage_HH);
```

```
                this->Controls->Add(this->label_mes_tran_frequency);
                this->Controls->Add(this->label_mes_tran_power);
                this->Controls->Add(this->label_tran_type);
                this->Controls->Add(this->label_mes_tran_type);
                this->Name = L"Tran";
                this->Text = L"Розрахунок роботи трансформатору";
                this->Load += gcnew System::EventHandler(this,
&Tran::Tran_Load);
                this->ResumeLayout(false);
                this->PerformLayout();

            }
#pragma endregion
      private: System::Void Tran_Load(System::Object^ sender,
System::EventArgs^ e) {
      }
      private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e) {

            //Перевірка на дурня
            if (comboBox_tran_induction->SelectedIndex > -1 &&
comboBox_tran_percent->SelectedIndex > -1 && comboBox_tran_plan-
>SelectedIndex > -1 &&
                comboBox_tran_power->SelectedIndex > -1 &&
comboBox_tran_stage->SelectedIndex > -1 && comboBox_tran_voltage_BH-
>SelectedIndex > -1 &&
                comboBox_tran_voltage_HH->SelectedIndex > -1) {
                //Проміжок коду для отримання значення діаметру описаної
окружності стержня магнітопроводу
                const double S_n = 0.35;

                int d_c = get_d_c(S_n);


                //Проміжок коду для отримання та виводу значення активного
перетину сталі
                const double K_z = 0.91;
                const double K_kr = 0.8 / 0.93;

                double S_a = get_S_a(K_z, K_kr, d_c);


                //Проміжок коду для отримання значення кількості витків в
обмотці НН
                int arr_with_induction_SIZE = 6;
                double* arr_with_induction = new
double[arr_with_induction_SIZE];
                double j = 1.40;
                for (int i = 0; i < arr_with_induction_SIZE; i++, j += 0.2)
                {
                      *(arr_with_induction + i) = j;
                }

                double B_c = arr_with_induction[comboBox_tran_induction-
>SelectedIndex];


                delete[]arr_with_induction;
                arr_with_induction = nullptr;
```

```
                int f = 50;


                int arr_with_voltage_HH_SIZE = 5;
                int* arr_with_voltage_HH = new
int[arr_with_voltage_HH_SIZE];
                for (int i = 0, k = 6300; i < arr_with_voltage_HH_SIZE;
i++, k += 200)
                {
                      *(arr_with_voltage_HH + i) = k;
                }

                int U_l1 = arr_with_voltage_HH[comboBox_tran_voltage_HH-
>SelectedIndex];


                delete[]arr_with_voltage_HH;
                arr_with_voltage_HH = nullptr;

                double U_1f;
                if (comboBox_tran_plan->SelectedIndex == 0) {
                      U_1f = get_U_1f(U_l1, 1.732);
                }
                else {
                      U_1f = get_U_1f(U_l1, 1.0);
                }

                int W_1 = get_W1(S_a, B_c, f, U_1f);

                //Проміжок коду для отримання значення ЕДС витку

                double E_v = get_E_v(U_1f, W_1);


                //Проміжок коду для отримання значення кількості витків в
обмотці ВН

                int arr_with_voltage_BH_SIZE = 5;
                int* arr_with_voltage_BH = new
int[arr_with_voltage_BH_SIZE];
                for (int i = 0, k = 3500; i < arr_with_voltage_BH_SIZE;
i++, k += 500)
                {
                      *(arr_with_voltage_BH + i) = k;
                }

                int U_l2 = arr_with_voltage_BH[comboBox_tran_voltage_BH-
>SelectedIndex];


                delete[]arr_with_voltage_BH;
                arr_with_voltage_BH = nullptr;

                double U_2f;
                if (comboBox_tran_plan->SelectedIndex == 0) {
                      U_2f = get_U_2f(U_l2, 1.732);
                }
                else {
```

```
                    U_2f = get_U_2f(U_l2, 1.0);
                }

                int W_2 = get_W2(U_2f, E_v);

                //Проміжок коду для отримання значень максимальної та
мінімальної кількості витків в обмотці ВН
                int arr_with_stages_SIZE = 4;
                int* arr_with_stages = new int[arr_with_stages_SIZE];
                for (int i = 0, k = 2; i < arr_with_stages_SIZE; i++, k +=
1)
                {
                    *(arr_with_stages + i) = k;
                }

                int n_ct = arr_with_stages[comboBox_tran_stage-
>SelectedIndex];


                delete[]arr_with_stages;
                arr_with_stages = nullptr;

                int arr_with_percent_SIZE = 6;
                double* arr_with_percent = new
double[arr_with_percent_SIZE];
                double k = 2.5;
                for (int i = 0; i < arr_with_percent_SIZE; i++, k += 0.5)
                {
                    *(arr_with_percent + i) = k;
                }

                double c_ct = arr_with_percent[comboBox_tran_percent-
>SelectedIndex];


                delete[]arr_with_percent;
                arr_with_percent = nullptr;

                int W_2_max = get_W2_max(W_2, n_ct, c_ct);
                int W_2_min = get_W2_min(W_2, n_ct, c_ct);

                //Проміжок коду для отримання значень перетину витків
обмоток НН та ВН

                double I_1f = get_I_1f(S_n, U_1f);
                double I_2f = get_I_2f(S_n, U_2f);

                const double sigma_1 = 2;
                const double sigma_2 = 3.5;

                double S_1v = get_S_1v(I_1f, sigma_1);
                double S_2v = get_S_2v(I_2f, sigma_2);

                //Проміжок коду для отримання значень висоти обмоток
                const double b_iz1 = 0.35;

                double H_1 = get_H1(S_1v, W_1, b_iz1);

                //Проміжок коду для отримання значення ширини обмотки ВН
```

```
                double b_iz2 = get_b_iz2(W_2_max, H_1);
                double a_2 = get_a_2(S_2v, W_2_max, H_1, b_iz2);

                //Проміжок коду для отримання значення напруги короткого
замикання
                const int a_01 = 20;
                const int a_1 = 0;
                const int a_12 = 20;

                double D_cp = get_D_cp(d_c, a_01, a_1, a_12);

                double t = get_t(a_1, a_2, a_12);

                double P_p = get_P_p(a_1, a_2, a_12, H_1);

                double U_k = get_U_k(I_1f, W_1, f, D_cp, t, P_p, E_v, H_1);

                //Проміжок коду для отримання значення перевишення
температури обмотки НН

                const double k_1 = 2.4;
                const double y = 0.0214;

                double P_1 = get_P_1(k_1, sigma_1, y, d_c, a_01, a_1, W_1,
S_1v);

                double Q_1_HH = get_Q_1_HH(P_1, d_c, a_01, a_1, H_1);

                const int a_v1 = 5;
                const int a_n1 = 10;
                const int t_m = 60;

                double t_1_HH = get_t_1_HH(Q_1_HH, a_v1, a_n1, t_m);

                //Проміжок коду для отримання значення перевишення
температури над маслом для обмотки катушки
                const int k_2 = 21;
                const int h_k1 = 3;
                double Q_1_oil = get_Q_1_oil(k_2, I_1f, sigma_1, W_1, a_1);

                double t_1_oil = get_t_1_oil(Q_1_oil, a_1, h_k1, a_v1,
a_n1, t_m);

                //Проміжок коду для отримання значення додаткових втрат в
обмотці
                const int N = 1;
                const int N_k = 1;
                const double b_k = 0.1;
                const int x = 1;

                double b_pr = get_b_pr(H_1, N, W_1 + W_2);
                double a_pr = get_a_pr(N, a_2, b_k, N_k, b_iz2);

                double S = get_S(a_pr, b_pr);

                double S_v = get_S_v(x, S);

                double sigma = get_sigma(I_2f, S_v);
```

```
                const double k_5 = 9.04;

                double B = get_B(0.96, I_1f, W_1, H_1);

                double P_add = get_P_add(k_5, sigma, a_pr, B, f);
                /*P_add = P_add * 100;*/

                //Проміжок коду для отримання значення втрат от циркулюючих
токів у результаті небездоганності транспозиції

                const double a_50 = 10.4;
                const int n_pr = 3;
                double a = get_a(f, a_50);

                double k_ob = get_k_ob(b_pr, H_1);

                double D_cp1 = get_D_cp1(d_c, a_01, a_1);
                double D_cp2 = get_D_cp2(d_c, a_01, a_1, a_12, a_2);
                double L = get_L((D_cp1 + D_cp2) / 2, (W_1 + W_2) / 2, x,
1);

                const double p = 0.0214;
                double r = get_r(p, L, S_v, 1, x);
                double p_1 = get_P_1(I_1f, r);
                double p_2 = get_P_2(I_2f, r, W_2, W_2_max);

                //Працює з частотою 50 гц
                double P_c = get_P_c(a_pr, b_iz2, k_ob, a, 1, n_pr, (p_1 +
p_2) / 2);

                //Проміжок коду для отримання значення критичної напруги по
умовам радіальної стійкості

                const double k1 = 1.2;
                const double k2 = 0.3;
                const double k8 = 1.4;
                double sigma_kr = get_sigma_kr(k8, k1, D_cp, k2, a_pr,
b_pr);

                //Проміжок коду для отримання значення плотності теплового
потоку

                const int P_k = 1;

                double q_v = get_q_v(P_k, a, D_cp, H_1);

                double k_z = get_k_z(D_cp, b_pr, n_pr);

                double q = get_q(q_v, a, k_z);

                //Проміжок коду для виводу усіх основних даних
                MessageBox::Show("Значення діаметру описаної окружності
стержня магнітопроводу : " + d_c.ToString() + "\n"
                        + "Значення активного перетину сталі : " +
S_a.ToString() + "\n"
                        + "Значення кількості витків в обмотці НН : " +
W_1.ToString() + "\n"
                        + "Значення ЕДС витку : " + E_v.ToString() + "\n"
```

```
                        + "Значення кількості витків в обмотці ВН : " +
W_2.ToString() + "\n"
                        + "Значення максимальної кількості витків в обмотці
ВН : " + W_2_max.ToString() + "\n"
                        + "Значення мінімальної кількості витків в обмотці ВН
: " + W_2_min.ToString() + "\n"
                        + "Значення перетину витків обмотки НН : " +
S_1v.ToString() + "\n"
                        + "Значення перетину витків обмотки ВН : " +
S_2v.ToString() + "\n"
                        + "Значення висоти обмоток : " + H_1.ToString() +
"\n"
                        + "Значення ширини обмотки ВН : " + a_2.ToString() +
"\n"
                        + "Значення напруги короткого замикання : " +
U_k.ToString() + "\n"
                        + "Значення перевишення температури обмотки НН : " +
t_1_HH.ToString() + "\n"
                        + "Значення перевишення температури над маслом для
обмотки катушки : " + t_1_oil.ToString() + "\n"
                        + "Значення додаткових втрат в обмотці : " +
P_add.ToString() + "\n"
                        + "Значення втрат от циркулюючих токів у результаті
небездоганності транспозиції : " + P_c.ToString() + "\n"
                        + "Значення критичної напруги по умовам радіальної
стійкості : " + sigma_kr.ToString() + "\n"
                        + "Значення плотності теплового потоку : " +
q.ToString() + "\n");

                  //Проміжок коду для запису данних до бази даних у вигляді
текстового файлу

                  values_to_file struct_data = get_values_to_struct(d_c, W_1,
W_2, W_2_max, W_2_min, S_a, E_v, S_1v, S_2v, H_1, a_2, U_k, t_1_HH, t_1_oil,
P_add, P_c, sigma_kr, q);

                  FILE* DATA_BASE = fopen("База даних.txt", "w");
                  if (DATA_BASE == NULL) {
                        MessageBox::Show("Помилка відкриття файлу!");
                        exit(1);
                  }

                  write_to_file(struct_data, DATA_BASE);


                  fclose(DATA_BASE);
            }
            else {
            MessageBox::Show("Заповніть дані!", "Помилка вводу",
MessageBoxButtons::OK, MessageBoxIcon::Error);

            }
      }
      private: System::Void label1_Click(System::Object^ sender,
System::EventArgs^ e) {
      }
      };
}
```