МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Литвиненко О.Є.
"_____"_____2022 р.

ДИПЛОМНИЙ ПРОЄКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Функціональна підсистема управління витратами авіаційного палива

Виконавець: _____ Прокопчук Д.Ю.

Керівник: _____ доц., к.ф.м.-н., Кучерява О.М.

Нормоконтролер: _____ доц., к.ф.м.-н., Кучерява О.М.

Київ 2022

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 126 «Інформаційні системи та технології»
(шифр, найменування)

Освітньо-професійна програма «Інформаційні системи та технології»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри
Литвиненко О.Є.
«_____»_____ 2022 р.

ЗАВДАННЯ
на виконання дипломної роботи (проєкту)
Прокопчука Дмитра Юрійовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проєкту): Функціональна підсистема управління витратами авіаційного палива
затверджена наказом ректора від 15.02.2022 р. № 251/ст .

2. Термін виконання роботи (проєкту): з 16.05.2022 р. по 19.06.2022 р.

3. Вихідні дані до роботи (проєкту): Мова програмування C#, середовище розробки Microsoft Visual Studio, декларативна мова розмітки XAML, платформа .NET Framework, платформа Universal Windows Platform.

4. Зміст пояснювальної записки:

1. Огляд існуючих функціональних систем управління витратами палива;

2. Технологічні засоби для розробки функціональної підсистеми управління витратами авіаційного палива;

3. Функціональна підсистема управління витратами авіаційного палива.

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1. Ілюстрація графічного інтерфейсу програми;

2. Схема алгоритму методу *Account_fuel()*.

Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Ознайомитись з матеріалами за темою дипломного проєкту.	16.05.2022 – 18.05.2022	
2	Оглянути і проаналізувати існуючі функціональні системи та технічні рішення. Підготувати текст першого розділу пояснювальної записки.	18.05.2022 – 21.05.2018	
3	Визначити функціонал та архітектуру функціональної підсистеми управління витратами авіаційного палива.	21.05.2022 – 25.05.2022	
4	Проаналізувати методи та засоби розробки функціональної підсистеми управління витратами палива. Підготувати текст другого розділу пояснювальної записки.	25.05.2022 – 30.05.2022	
5	Розробити алгоритми, написати додаток та протестувати програмний код. Підготувати текст третього розділу пояснювальної записки.	30.05.2022 – 05.06.2022	
6	Завершити оформлення пояснювальної записки. Оформити графічний матеріал.	06.06.2022 – 11.06.2022	
7	Підготувати доповідь та презентацію.	11.06.2022 – 16.06.2022	
8	Захист дипломного проєкту.	18.06.2022	

7. Дата видачі завдання: « 16 » травня 2022 р.

Керівник дипломної роботи (проєкту) _____ Кучерява О.М.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Прокопчук Д.Ю.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту «Функціональна підсистема управління витратами авіаційного палива» містить 45 сторінок, 16 рисунків, 13 використаних джерел.

ЛІТАК, ПАЛИВО, ПАЛИВНА СИСТЕМА, СЕРЕДОВИЩЕ РОЗРОБКИ *MICROSOFT VISUAL STUDIO*, МОВА ПРОГРАМУВАННЯ *C#*, ПЛАТФОРМА *.NET FRAMEWORK*, ПЛАТФОРМА *UNIVERSAL WINDOWS PLATFORM*.

Об’єкт дипломного проєктування – розхід авіаційного палива.

Предмет дипломного проєктування – підсистема управління витратами авіаційного палива.

Мета дипломного проєкту – розробка функціональної підсистеми управління витратами авіаційного палива.

Методи проєктування – сучасні методи розробки електронних пристроїв, методи розробки підсистем управління витратами палива, мова програмування *C#*, декларативна мова розмітки XAML, середовище розробки *Microsoft Visual Studio*, платформа *.NET Framework*, платформа *Universal Windows Platform*.

Результатом виконання дипломного проєкту є розроблена функціональна підсистема управління витратами авіаційного палива, призначена для розрахунку маси палива та його об’єму. Оброблення цієї інформації, видачі керуючих сигналів та індикації розходу палива.

Результати дипломного проєктування рекомендується застосувати у авіаційній промисловості.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

ЛА	Літальний апарат
АРП	Автомат розподілу палива
ПС	Паливна система
АСК	Автоматизована система контролю
БПР	Бортовий пристрій реєстрації
CLR	*Common Language Runtime*
CTS	*Common Type System*
UWP	*Universal Windows Platform*
VS	*Visual Studio*
XAML	*Extensible Application Markup Language*
XML	*Extensible Markup Language*

ВСТУП

Актуальність теми. Україна за рівнем розвитку літакобудування, належить до найбільш розвинених держав. Саме наша авіабудівна галузь має повний цикл створення авіаційної техніки, та посідає провідне місце на світовому ринку в секторі транспортної та регіональної пасажирської авіації.

На сьогодні існує багато сучасних літаків та паливних систем для них. Вони змінюються та покращуються, щоб забезпечити автоматизацію та підвищити людську безпеку. В літаках паливні системи призначені для розміщення палива та безкінечної подачі його до двигунів у необхідній кількості та з достатнім тиском на всіх заданих режимах та висотах польоту. Провідну роль займає, звичайно, такий чинник, як об’єм палива, який витрачається.

Витрата палива літака є основним показником, що відображає ефективність експлуатації повітряного судна. Адже, чим нижча витрата палива для певної моделі, тим менше витрат приносить його експлуатація авіакомпанії. На сучасних літаках об’єм палива може сягати багатьох десятків тон. При польотах на значні відстані паливо розміщують у великій кількості баків, що встановлюються у крилі та рідше у фюзеляжі. Найчастіше застосовуються три типи паливних баків: жорсткі, м’які та герметичні баки-відсіки.

Спільне застосування паливних систем обумовлено необхідністю вимірювати не тільки запас, але й витрати палива, яке поступає до двигунів літака. Відомо, що наявність на борту літального апарату лише витратоміра не гарантує точного визначення витрат палива, а залишку у разі витоку палива з паливної системи й поготів. Але наявність на борту паливоміру та декількох приладів витратоміра збільшує загальну масу обладнання, кількість візуальних приладів та ускладнює роботу пілотові. У зв’язку з цим в останні рокі намітилася тенденція до створення комбінованих систем управління витратами палива. Дані системи працюють на один прилад, який показує паливомірні та витратомірні результати. Це дозволило отримати виграш у масі, забезпечити точне вимірювання запасу палива в особливих аварійних ситуаціях різних еволюцій літаків, і навіть спростило індикацію поточних значень запасу та витрат палива.

Мета і завдання виконання дипломного проекту. Метою дипломного проєкту є розробка функціональної підсистеми управління витратами авіаційного палива. Використання даної підсистеми дозволить:

- розраховува об'єм палива;
- розраховува масу палива;
- відслідковува вільний обсяг палива;
- відслідковува загальну місткість бака;
- визначити площу передньої поверхні бака;
- визначити площу бічної поверхні бака;
- визначити площу загальної ємності.

Відповідно до мети дипломного проєкту, було визначено такі завдання:

- огляд існуючих функціональних систем та технічних рішень управління витратами палива;
- розробка та опис предметної області інформаційно-вимірювальної системи розходу авіаційного палива;
- проєктування та розробка алгоритму роботи функціональної підсистеми управління витратами палива;
- розробка програмного коду на мові *C#*;
- демонстрація функціонування створеного додатка.

Об’єкт і предмет проєктування. Об’єктом дипломного проєктування є розхід авіаційного палива.

Предметом дипломного проєктування являється підсистема управління витратами авіаційного палива.

Вимірювання маси палива базується на принципі вимірювання ємності датчиків, встановлених у паливних баках. Маса пального вимірюється з урахуванням діелектричної проникності діелектрика ємнісного датчика, на який впливають багато чинників.

Підсистема буде в складі основної паливної системи управління витратами авіаційного палива.

Методи проєктування. Щоб досягти виконання зазначених завдань дипломного проєктування було вирішено застосувати навички по роботі в інтегрованому середовищі розробки *Microsoft Visual Studio*. Дане середовище було обрано через його здатність об'єднувати процеси розробки та підтримувати велику кількість мов програмування і різних технологій.

Мовою програмування для реалізації алгоритмів було обрано *C#*. Мова була вибрана через велику кількість підтримуваних технологій, бібліотек та можливість легко прокомпілювати підсистему на інші операційні системи.

Для розробки також було застосовано платформу *Universal Windows Platform*. Вона використовується для створення універсальних програм, що запускаються на *Windows 10*, без зміни у коді. Підтримує обрану мову програмування *C#*.

Для збільшення функціоналу мови програмування *C#* було обрано також платформу *.NET Framework*. Ця технологія підтримує створення і виконання нового покоління додатків. Обрана платформа *Universal Windows Platform* легко поєднується з *.NET Framework*.

При роботі з платформами *Universal Windows Platform* та *.NET Framework* було застосовано декларативну мову розмітки *XAML*. Дана мова розмітки забезпечує робочий процес, який дозволяє легко розробляти користувацький інтерфейс і логіку програми, використовуючи потенційно різні засоби.

Практичне значення отриманих результатів. У даному дипломному проєкті розроблена функціональна підсистема управління витратами авіаційного палива призначена для вимірювання маси палива та його об’єму. Оброблення цієї інформації, видачі керуючих сигналів та індикації розходу палива. Галузь використання – авіаційна промисловість.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ ФУНКЦІОНАЛЬНИХ СИСТЕМ УПРАВЛІННЯ ВИТРАТАМИ ПАЛИВА

1.1 Сучасні технічні рішення щодо витрат палива

В наш час вже існує безліч приладів, які вимірюють об'ємну чи вагову кількість палива у баках літальних апаратів. Вони дозволяють екіпажу літака будь-якої миті польоту визначити, скільки палива знаходиться в баку, та розрахувати час, упродовж якого літак ще може знаходитись в небі.

Звичайно, безпосереднє вимірювання обсягу палива на борту літака неможливе, тому застосовують методи непрямого вимірювання, в яких обсяг палива функціонально пов'язаний з будь-якою величиною, що легко визначається. Як еталон таких величин зазвичай вибирають об’єм палива в баку.

Завдяки сучасним системам управління витратами авіаційного палива можна визначити сумарний запас палива у всіх баках та кількість палива в кожному з них окремо. Необхідно знати, як пальне розподілене між баками, щоб визначити правильну послідовність витрачання палива з баків та уникнути неприпустимого зміщення центру мас літака. Саме автоматичні пристрої управління витратами авіаційного палива керують перемиканням баків.

Більшість методів виміру кількості палива зводиться до виміру його рівня (висоти стовпчика рідини). Однак системи управління витратами палива градуюють в одиницях об'єму (літрах) або кілограмах. Тому нанесення шкали відповідності між певними показниками залежить від розмірів та типів паливних баків, котрі вимірюються даним приладом.

Також паливо-вимірювальні комплекси крім виконуваних ними завдань вимірювання палива та управління витратами передбачають широкі зв'язки з бортовими пристроями реєстрації (БПР), автоматизованими системами контролю (АСК) і наземними пунктами управління польотами, видають інформацію про дальність і тривалість польоту в навігаційні пілотні комплекси.

Розподіляючи дані системи за принципом дії, можна відзначити типи, що набули найбільшого поширення:

- поплавкові, засновані на вимірі рівня пального за допомогою плаваючого на поверхні поплавця;
- манометричні, засновані на вимірі тиску стовпа палива за допомогою манометра;
- ємнісні, засновані на вимірі рівня пального спеціальним конденсатором.

Системи управління витратами авіаційного палива бувають двох видів: електричні та механічні. Дані системи повинні бути дистанційними, тому зазвичай в авіації використовують електричний вид. Механічні системи, не будучи дистанційними, майже не застосовуються в авіації.

1.2 Поплавкові системи вимірювання розходу авіаційного палива

Вимірювання запасу палива в баку літального апарата за допомогою електричного поплавкового паливоміра базується на принципі перетворення неелектричної величини (змінна висота рівня палива) в електричну величину (змінний активний опір), мінливу відповідно до зміни рівня рідини. Це перетворення відбувається за рахунок реостатних датчиків поплавкового типу, які встановлені в баки літака. Покажчиком служить магнітоелектричний логометр.

Авіаційні електричні поплавкові системи поділяються на паливоміри за видом авіаційного палива, за типом електросхем, наявністю або відсутністю сигналізації і мають відповідне маркування. Буквене маркування датчиків фіксує

витрати: Б – бензину, К – керосину, М – масла. Вимір запасу бензину - БЕ, вимір запасу керосину - КЕ , вимір запасу масла - МЕ.

Робота автоматичної частини поплавкової системи вимірювання відбувається таким чином, що при досягненні потрібного рівня поплавок передає дані на сигналізатор, який в свою чергу замикає контакти сигнального пристрою. Даний пристрій подає сигнал на агрегати, що управляють витратою або заправкою, і на сигнальні лампи.

Найпростішим із розглянутої групи є паливомір БЕ-46. Цей прилад призначений для індивідуального контролю кількості палива в баку літака. В якості вказівника у ньому застосовується магнітоелектричний логометр з рухомими котушками, розташованими під кутом один до одного. Завдяки цьому, а також завдяки оригінальній конструкції магнітної системи розмах шкали логометра становить 180°. Одним з основних вимог до конструкції датчика даного приладу є повний захист камери потенціометра від проникання в неї парів палива, наявність яких в датчику може призвести до вибуху.

У зв'язку з цим довелося відмовитися від зубчастої передачі між віссю повзунка потенціометра та важелем поплавця. Забезпечити мале тертя і водночас високу герметичність у місці входу осі повзунка до камери реостата завдання досить складне. Набагато простіше забезпечити герметичність камери, якщо передачу руху від поплавка до повзунка реостата здійснити за допомогою системи важелів. Саме так і виконана система передачі в датчику БЕ-46, конструктивна схема якого зображена на рис. 1.1.

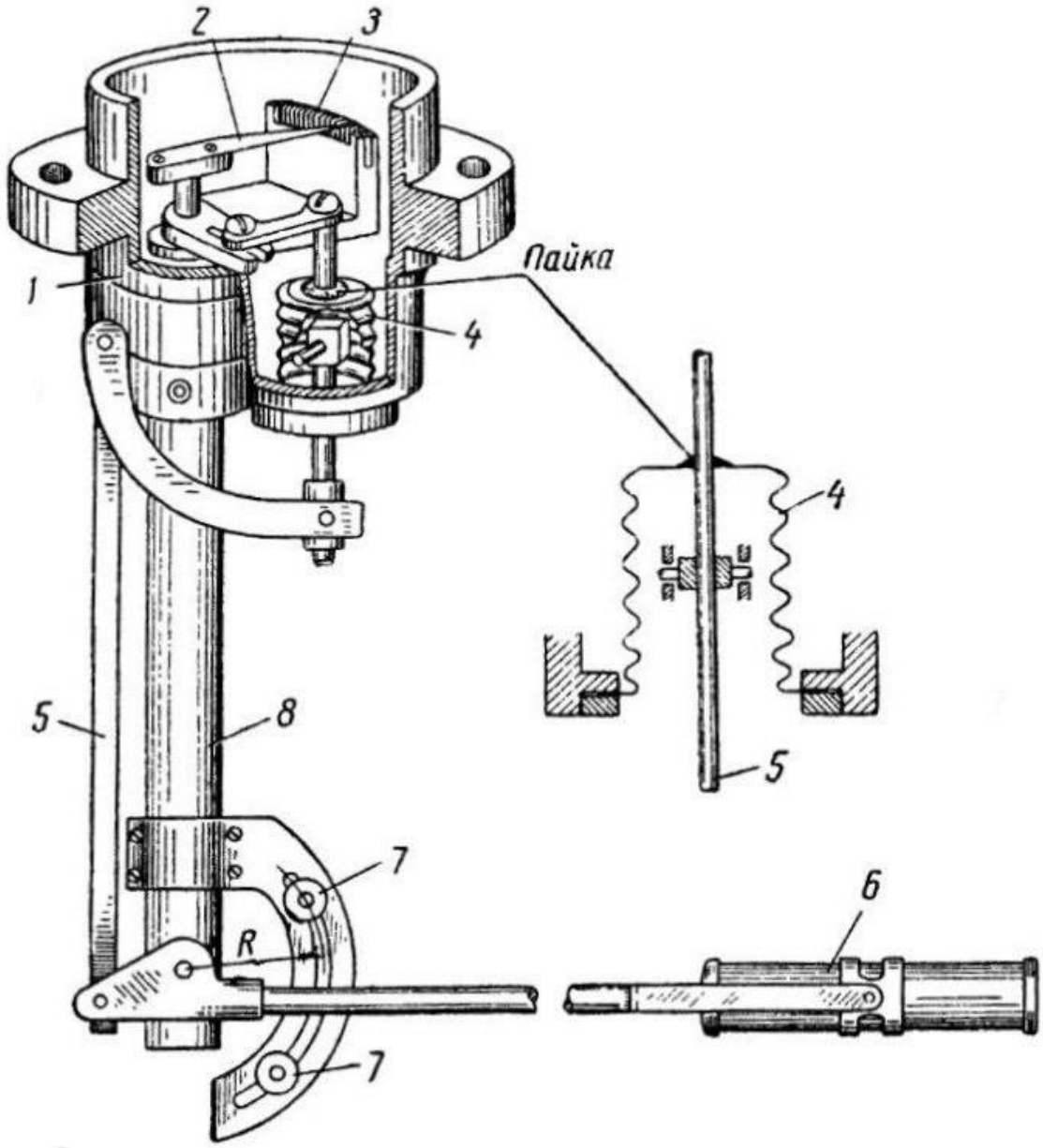

Рис. 1.1. Конструктивна схема поплавкового датчика БЕ-46

1 - корпус потенціометра датчика; 2 - повзунок потенціометра; 3 - потенціометр; 5 - важіль передавального механізму; 6 - поплавок; 7 - обмежувач; 8 - колонка.

Важіль 5 , що входить в камеру потенціометра датчика, впаюється в гофровану еластичну мембранну коробку (сильфон) 4. Сильфон допускає хитання важеля 5 і в той же час завдяки глухому з'єднанню основи сильфона з корпусом

потенціометра забезпечується герметичність введення передавального механізму в корпус.

Як вже зазначалося, паливоміри кожного типу можуть бути градуйовані порізному. У БЕ-46, як і у всіх інших приладів, одне градуювання відрізняється від іншої висотою колонки 8, довжиною важеля поплавця 6 і профілем потенціометра. Різні висота колонки і довжина важеля поплавця залежать від розмірів баків, для встановлення яких і призначений прилад.

Зв'язок між датчиком і вказівником БЕ-46 здійснюється за допомогою найпростішої потенціометричної дистанційної передачі на постійному струмі (рис. 1.2.).

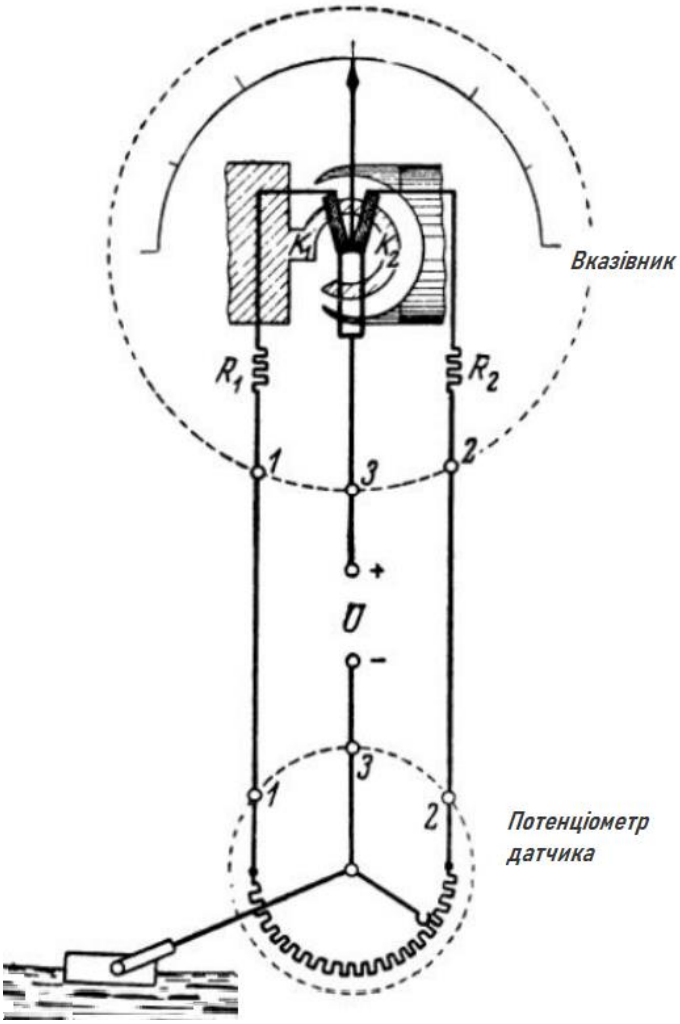

Рис. 1.2. Принципова електрична схема паливоміру БЕ-46

K1, K2 - котушки логометра; *R1, R2* - підганяльні опори; 1, 2, 3 - клеми приєднання живлення і електропроводів лінії зв'язку.

В польоті пілоту дуже важливо вчасно дізнатися, що в баку літака залишився обмежений (так званий критичний) запас палива, мінімально необхідний для успішного завершення польоту. Ці дані можна отримати шляхом періодичного спостереження за показаннями датчика, однак це стомлює і не завжди можливо. Тому, як правило, паливоміри доповнюються спеціальним пристроєм, який сигналізує про критичний залишок палива в баках літака. Сигналізація здійснюється зазвичай загоранням червоної лампочки, встановленої на передній панелі в кабіні екіпажу біля вказівника паливоміра.

1.3 Ємнісні системи вимірювання розходу авіаційного палива

Принцип дії ємнісного датчика заснований на залежності величини ємності спеціального конденсатора від рівня палива в баку. Чутливий елемент ємнісного паливоміра (рис. 1.3.) являє собою циліндричний конденсатор з внутрішнім та зовнішнім електродами та ізоляційним шаром. Між ізоляційним шаром і зовнішнім електродом знаходиться шар палива, рівень якої необхідно виміряти. Якщо рівень рідини в баку змінюється, то буде змінюватися і ємність конденсатора внаслідок того, що діелектричні постійні рідини і повітря різні.

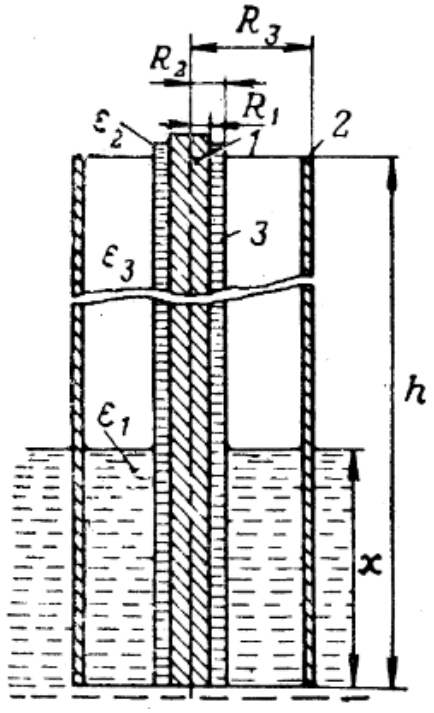

Рис. 1.3. Принципова схема чутливого елемента ємнісного паливоміра

1 - внутрішній електрод; 2 - зовнішній електрод; 3 - ізоляційний шар; ε_1, ε_2, ε_3 - діелектричні постійні рідини, матеріалу ізолятора та суміші парів рідини і повітря відповідно; R_1, R_2, R_3 - радіуси внутрішнього електрода, ізолятора і зовнішнього електрода; x - рівень рідини; h - повна висота датчика.

У більшості випадків зовнішній електрод циліндричного конденсатора повинен бути виконаний окремо, проте не виключено використання в якості зовнішнього електрода стінок бака, особливо у високих і вузьких баках. Це тим більш доцільно, що в такому випадку конденсатор дозволяє вимірювати кількість палива в баку без помітних похибок при досить великих кренах літака і прискореннях.

Ємнісні паливоміри застосовуються для вимірювання кількості всіх видів палива, але виявляються майже незамінними у разі вимірювання кількості хімічно активних рідин, застосовуваних в якості горючих компонентів у рідинно-реактивних двигунах. У цьому випадку внутрішню трубку конденсатора датчика покривають тонким електроізоляційним шаром. Матеріал для зовнішньої трубки також слід вибирати з урахуванням властивостей рідини, рівень якої потрібно вимірювати.

Якщо знехтувати кінцевим ефектом, то можна прийняти, що ємність нижній частині циліндричного конденсатора буде:

$$C_x = \frac{x}{2\left|\frac{1}{\varepsilon_2} ln\frac{R_2}{R_1} + \frac{1}{\varepsilon_1} ln\frac{R_2}{R_1}\right|} \qquad (1.1)$$

Подібно до цього ємність верхній частині конденсатора знайдемо із співвідношення:

$$C_x = \frac{h - x}{2\left|\frac{1}{\varepsilon_2} ln\frac{R_2}{R_1} + \frac{1}{\varepsilon_3} ln\frac{R_3}{R_2}\right|} \qquad (1.2)$$

Підсумовуючи ємності C_x і C_h, отримаємо повну ємність конденсатора:

$$C = \frac{x}{2}\left|\frac{1}{\left|\frac{1}{\varepsilon_2} ln\frac{R_2}{R_1} + \frac{1}{\varepsilon_1} ln\frac{R_2}{R_1}\right|} - \frac{1}{\left|\frac{1}{\varepsilon_2} ln\frac{R_2}{R_1} + \frac{1}{\varepsilon_3} ln\frac{R_3}{R_2}\right|}\right| + \\ + \frac{h}{2}\frac{1}{\left|\frac{1}{\varepsilon_2} ln\frac{R_2}{R_1} + \frac{1}{\varepsilon_3} ln\frac{R_3}{R_2}\right|} \qquad (1.3)$$

З цього виразу випливає, що ємність конденсатора є лінійною функцією рівня рідини *x*. Таким чином, вимірювання рівня рідини можна звести до вимірювання ємності конденсатора *C*.

Чутливість ємнісного датчика визначається виразом:

$$\frac{dC}{dx} = \frac{1}{2}\left|\frac{1}{\left|\frac{1}{\varepsilon_2} ln\frac{R_2}{R_1} + \frac{1}{\varepsilon_1} ln\frac{R_3}{R_1}\right|} - \frac{1}{\left|\frac{1}{\varepsilon_2} ln\frac{R_2}{R_1} + \frac{1}{\varepsilon_3} ln\frac{R_3}{R_2}\right|}\right| \qquad (1.4)$$

Легко бачити, що найбільша чутливість буде в тому випадку, коли R_2/R_1 прагне до 1, тобто коли шар ізоляції відсутній. При цьому отримаємо:

$$\frac{dC}{dx} = (\varepsilon_1 - \varepsilon_3) ln\left(\frac{R_2}{R_3}\right)^{\frac{1}{2}} \qquad (1.5)$$

Оскільки діелектрична стала напівпровідних рідин значно більша, ніж непровідних, то зміна ємності на одиницю довжини в першому випадку буде більшою, ніж у другому. Звідси випливає, що ємнісний метод вимірювання рівня є особливо ефективним для напівпровідних рідин.

З формули (1.5) випливає, що для збільшення чутливості немає необхідності брати велику величину R_3/R_2. Якщо величина $R_3 – R_2$ мала, то на точність показань приладу значний вплив буде давати в'язкість рідини. Отже, шар рідини між електродами повинен бути таким, щоб в'язкість не впливала на рівень рідини. Зазвичай обмежуються зазором $R_3 – R_2$ (l,5 – 6 мм), а для збільшення чутливості датчик збирають з декількох концентричних труб, що утворюють паралельно з'єднані конденсатори.

На рис. 1.4. зображено контур циліндричного, спрофільованого датчика ємнісної системи вимірювання палива.

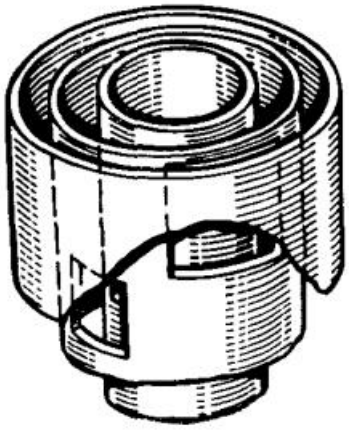

Рис. 1.4. Контур циліндричного, спрофільованого датчика ємнісної системи вимірювання палива

Слід зазначити, що в ємнісному паливомірі можна повністю компенсувати методичні похибки, що виникають від нахилу бака при крені та прискореннях. Дійсно, для цього достатньо замість одного чутливого елемента встановити по краях бака чотири елементи. При паралельному з'єднанні чутливих елементів загальна ємність їх залишатиметься майже постійною за будь-яких нахилів бака.

1.4 Манометричні системи вимірювання авіаційного палива

Манометричні системи вимірювання палива засновані на використанні деформації пружного чутливого елемента, при впливі на нього тиску, що вимірюється. Сам датчик призначений для безперервного перетворення надлишкового тиску в уніфікований вихідний сигнал змінного струму, заснований на зміні взаємної індуктивності. Зазвичай прилад встановлюється під баком (в стаціонарних умовах) для роботи в комплексі з вторинними взаємозамінними диференціально-трансформаторними приладами, машинами централізованого контролю та іншими приймачами інформації, здатними приймати стандартний сигнал у вигляді взаємної індуктивності.

На рис. 1.5. зображена трубчаста пружина, яка служить перетворювачем манометричної системи вимірювання МЕД.

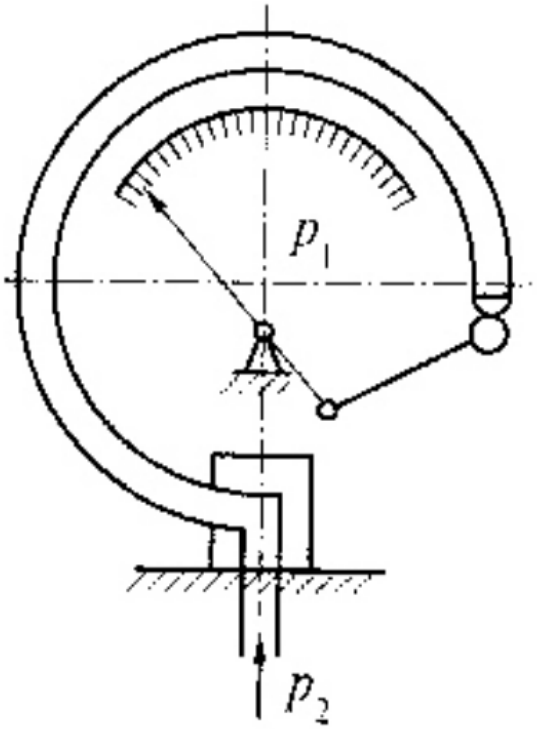

Рис. 1.5. Чутливий елемент манометра МЕД

Трубчаста пружина є тонкостінною, зігнутою по дузі кола, трубкою, з напаяним одним кінцем, яка виготовляється з мідних сплавів або нержавіючої сталі. При збільшенні чи зменшенні тиску всередині трубки пружина розкручується чи скручується на певний кут.

Саму трубку в приладі розташовують так, щоб мала вісь перерізу лежала в порожнині згину трубки. При заповненні порожнини трубки рідиною під тиском відбувається деформація перерізу в напрямку наближення до кругу, це викликає появу зусиль, які змушують трубку розгинатися. Таким чином, у пружинних приладах використовується властивість спіральної трубки розкручуватися зі збільшенням тиску всередині неї і стискатися при зменшенні тиску.

На рис. 1.6. зображено пристрій найпростішого пружинного манометра МЕД.

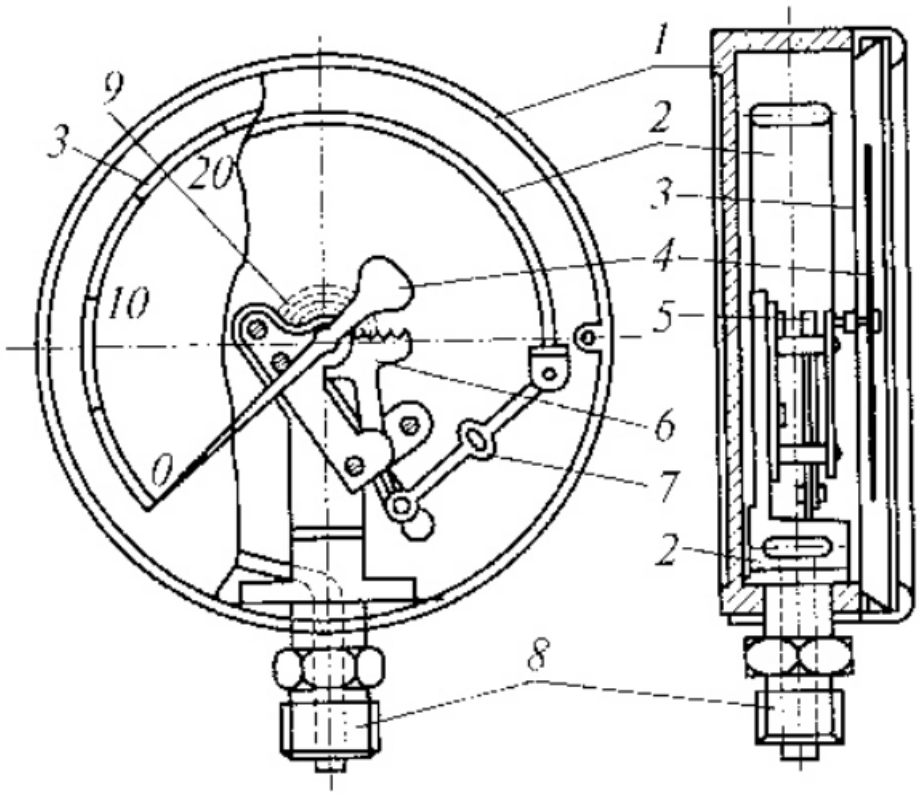

Рис. 1.6. Пружинна манометрична система вимірювання МЕД

1 - корпус; 2 - порожниста трубка; 3 - вимірювальна шкала; 4 - вказівна стрілка; 5 - маленька шестерня, 6 - зубчастий сектор; 7 - повідець; 8 - штуцер; 9 - спіральна пружинка.

Чутливий елемент приладу, що сприймає сигнал тиску, виконаний у формі зігнутої по колу порожнистої трубки 2 під кутом 270° з поперечним перетином у вигляді еліпса або плоского овалу. Один кінець трубки вільний і наглухо закритий, а інший кінець її впаяний у тримач, який приєднується до джерела вимірюваного тиску за допомогою штуцера 8. Закритий кінець трубки повідцем 7 з'єднаний із зубчастим сектором 6, який зчеплений з маленькою шестернею 5, що сидить на одній осі з вказівною стрілкою 4. Під дією надлишкового тиску трубка згинається, закритий (вільний) кінець трубки переміщується і тягне повідець 7, який повертає пов'язаний з ним зубчастий сектор 6. Переміщаючись, зубчастий сектор обертає маленьку шестерню 5 з насадженою на її вісь стрілкою, що вказує за шкалою 3 величину вимірювального тиску. Щоб уникнути мертвого ходу між зубцями

сектора та шестернею, застосована спіральна пружинка 9, що притискаю малу шестерню до одного боку зубців сектора. Всі вказані елементи змонтовані в корпусі 1.

1.5 Висновки до розділу

В даному розділі було розглянуто існуючі функціональні системи управління витратами авіаційного палива. Ці системи бувають двох видів: електричні та механічні. Оскільки на сьогодні механічні системи є непрактичними, вони майже не використовуються в авіації. Для пілотів та інших працівників обслуговування літальних апаратів, куди зручніше вимірювати паливо та контролювати ці витрати дистанційно. Тому зазвичай в авіації використовують електричний вид системи.

За принципом своєї дії системи управління витратами палива можна поділити на типи, які набули найбільшого поширення:

- поплавкові, засновані на вимірі рівня пального за допомогою плаваючого на поверхні поплавця;
- манометричні, засновані на вимірі тиску стовпа палива за допомогою манометра;
- ємнісні, засновані на вимірі рівня пального спеціальним конденсатором.

Розглянуті паливоміри, що входять до систем управління витратами авіаційного палива, мають різні модифікації і виконують наступні функції:

- вимірюють кількість палива у групах баків та сумарну кількість палива на літаку;
- керують виробленням палива за заданою програмою;
- здійснюють управління заправкою палива;
- сигналізують про вироблення палива з певної групи баків та про залишок палива на певну тривалість польоту;
- деякі з них системи здійснюють, крім того, автоматичне центрування літака.

РОЗДІЛ 2
ТЕХНОЛОГІЧНІ ЗАСОБИ ДЛЯ РОЗРОБКИ ФУНКЦІОНАЛЬНОЇ ПІДСИСТЕМИ УПРАВЛІННЯ ВИТРАТАМИ АВІАЦІЙНОГО ПАЛИВА

2.1 Середовище розробки *Microsoft Visual Studio*

Microsoft Visual Studio – це продукт компанії *Microsoft*, який включає в себе інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, у тому числі з підтримкою технології *Windows Forms*, а також веб-сайти, веб-програми, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних *Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.* Крім стандартного редактора і відладчика, які є в більшості середовищ *IDE, Visual Studio* включає компілятори, засоби автозавершення коду, графічні конструктори і багато інших функцій для поліпшення процесу розробки. До *Visual Studio* входять такі компілятори, як: *Visual Basic, Visual Basic .NET, Visual C++, Visual C#, Visual F#, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, Visual Source Safe*.

Користувачу надається можливість також обрати версію даного середовища: *Visual Studio Community, Visual Studio Professional* та *Visual Studio Enterprise*.

1. *VS Community* – це безкоштовна версія даного середовища, яка дуже наближена до професійної версії. Розробники можуть використовувати її для створення сучасних програм *Android, iOS* та *Windows*, а також веб-застосунків та хмарних служб. Для завантаження доступні інструменти *Xamarin* і *Unity*, що

дозволяють створювати, налагоджувати, тестувати, а також спільно та повторно використовувати код для більшої кількості платформ. Також серед безкоштовних інструментів, користувачам надається можливість розробляти веб-додатки за допомогою *ASP.NET, Node.js, Python* та *JavaScript*. Використовувати потужні веб-платформи, такі як: *AngularJS, jQuery, Bootstrap, Django* та *Backbone.js*.

Visual Studio Community може використовувати необмежену кількість користувачів в організації у таких випадках: у навчальних аудиторіях, для наукових досліджень або участі у проектах з відкритим кодом. Для всіх інших сценаріїв використання, тобто у некорпоративних організаціях Visual Studio Community можна використовувати до 5 користувачів.

2. *VS Professional* – це платна версія даного середовища для індивідуальних розробників чи невеликих команд. *VS Professional* забезпечує ефективні функції, що дозволяють швидко розуміти код. *CodeLens* допомагає зосередитися на роботі, відображаючи посилання на код та зміни в коді, показуючи, хто вносив у метод останні зміни, і визначаючи, чи успішно виконуються тести, і все це при безпосередній роботі з кодом. Також в даній версії є можливість напряму користуватися *Server Explorer* та з’явилась інтеграція з *Microsoft SQL Server*.

До складу підписки входить *Azure DevOps* – набір служб для планування та створення програм, а також швидкої доставки цих програм до будь-якого хмарного або локального розташування. У розпорядженні розробників гнучкі засоби планування, платформа безперервної інтеграції та постачання, система управління версіями та репозиторій артефактів.

Не кажучи про продукт *GitHub Enterprise*. В *VS Professional* він надає переваги, пов'язані, серед іншого, з автоматичним перенесенням коду в хмару, а також використанням відкритого коду та комплексної автоматизованої системи безпеки, щоб користувачі могли швидше створювати інновації та впевнено доставляти їх.

3. *VS Enterprise* – це також платна версія даного середовища для команд будь-якого розміру з високими вимогами до якості та масштабу. В даній версії є можливість здійснювати раннє тестування та прискорювати роботу над кодом за

допомогою функцій реального часу: *IntelliTest, Live Unit Testing* та динамічні перевірки залежності. Доступне розширене налагодження та діагностика, корпоративні методики розробки та операцій.

Microsoft Visual Studio включає редактор вихідного коду з підтримкою технології *IntelliSense* і можливістю найпростішого рефакторингу коду.

IntelliSense – представляє собою набір можливостей, що відображають відомості про код безпосередньо в редакторі і в деяких випадках автоматично створюють невеликі уривки коду. По суті, це вбудована в редактор базова документація, яка позбавляє необхідності шукати інформацію в інших джерелах. На рис. 2.1. зображено, як *IntelliSense* відображає список членів типу.

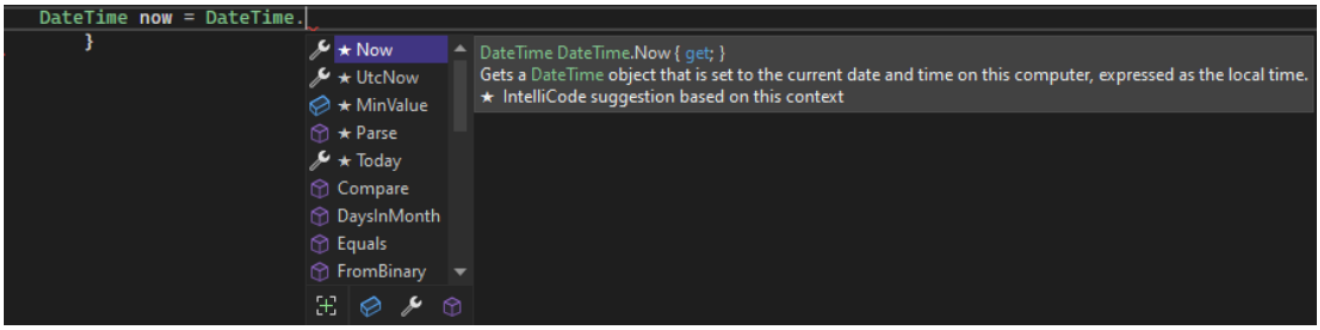

Рис. 2.1. Робота технології *IntelliSense*

Рефакторинг коду включає в себе такі операції, як інтелектуальне перейменування змінних, вилучення однієї або кількох рядків коду в новий метод і зміна порядку параметрів методів (рис. 2.2.).

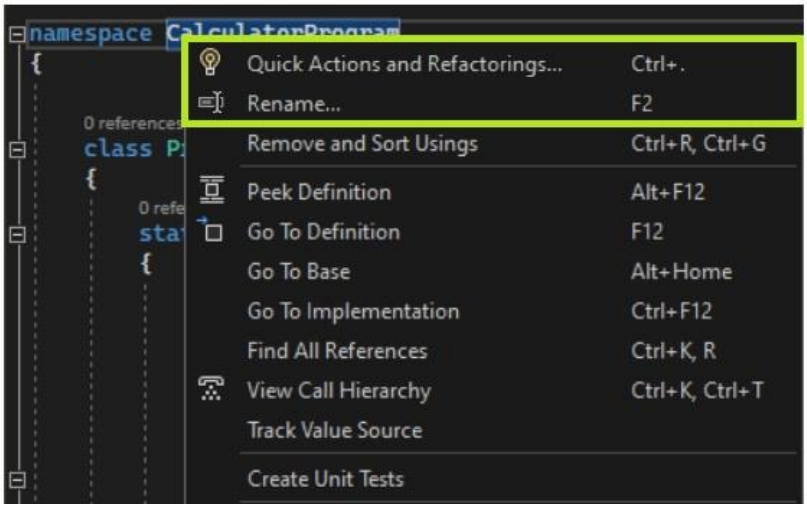

Рис. 2.2. Рефакторинг коду

Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і як відладчик машинного рівня. Інші вбудовані інструменти включають редактор форм для спрощення створення графічного інтерфейсу програми, веб-редактор, дизайнер класів і дизайнер схеми бази даних.

Visual Studio дозволяє створювати та підключати сторонні доповнення (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, *Subversion* та *Visual SourceSafe*), додавання нових наборів інструментів, наприклад, для редагування та візуального проектування коду на предметно-орієнтованих мовах програмування або інструментів інших аспектів циклу розробки програмного забезпечення (наприклад, клієнт *Team Explorer* для роботи з *Team Foundation Server*).

На рис. 2.3. зображено середовище *Microsoft Visual Studio* з відкритим проектом та підказками по основним вікнам та функціональним можливостям.

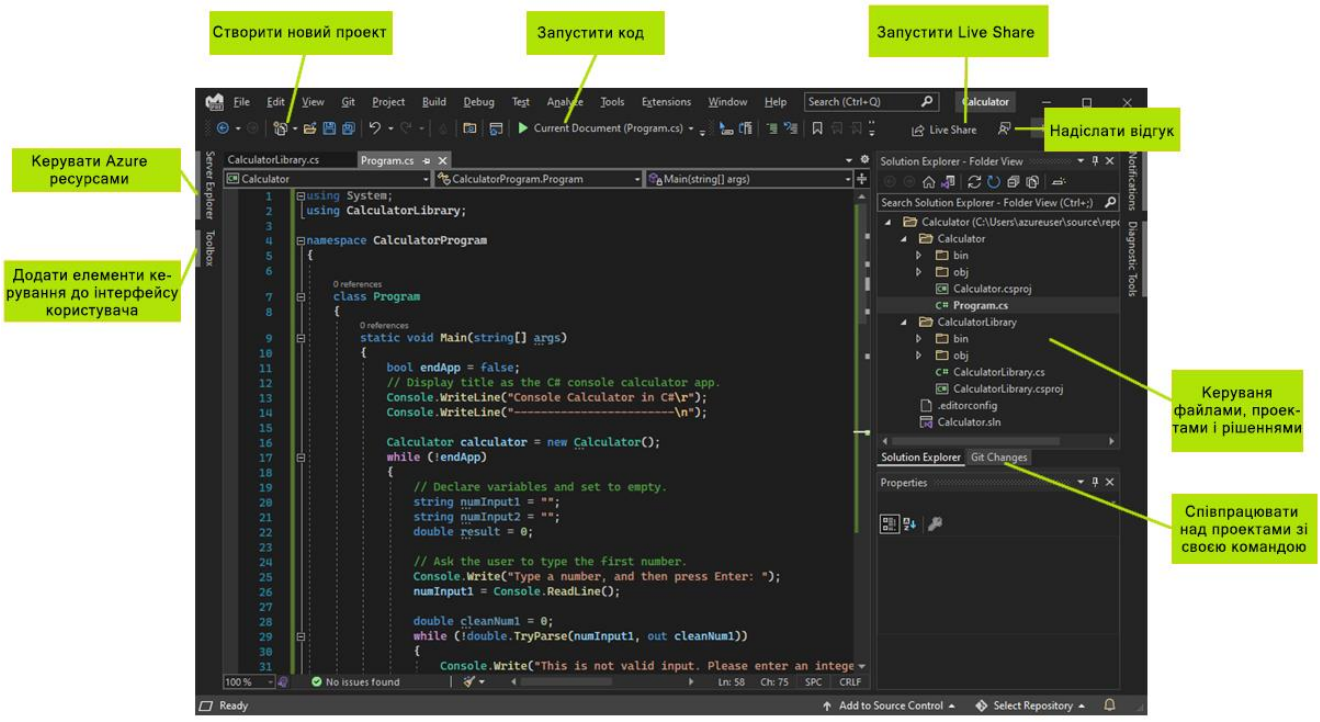

Рис. 2.3. Відкритий проект в середовищі *Microsoft Visual Studio*

Праворуч у верхньому куті вікна від Оглядача рішень можна переглядати файли коду, переміщатися по них і керувати ними. Оглядач рішень дозволяє впорядкувати код шляхом об'єднання файлів у рішення та проекти.

У центральному вікні редактора, з яким користувачі, ймовірно, працюють найдовше, відображається вміст файлу. У вікні редактора можна вносити зміни в код або розробляти інтерфейс користувача, наприклад вікно з кнопками або текстові поля.

Вікно змін Git у нижньому куті праворуч дозволяє відстежувати робочі елементи та надавати спільний доступ до коду за допомогою Git, GitHub або інших технологій керування версіями.

2.2 Мова програмування *C#*

C# (вимовляється Сі-шарп) – проста, сучасна об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи *.NET*. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою *Microsoft Research*. Дана мова програмування є однією з найпотужніших, швидко розвинених і бажаних мов в IT-галузі. На сьогодні на ній пишуться самі різні програми: від невеликих комп'ютерних програмок до великих веб-порталів і веб-сервісів, що обслуговують щодня мільйони користувачів.

C# є мовою із С-подібним синтаксисом і близькою у цьому відношенні до *C++* та *Java*. Тому якщо користувач знайомий хоча б з однією з цих мов, то оволодіти *C#* йому буде легше. *C#* підтримує поліморфізм, успадкування, навантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих та розширюваних додатків.

Крім того, що *C#* є об'єктно-орієнтованою мовою, вона також підтримує і компонентно-орієнтоване програмування. Розробка сучасних програм все більше тяжіє до створення програмних компонентів у формі автономних та самоописових пакетів, що реалізують окремі функціональні можливості. Важливою особливістю

таких компонентів є модель програмування на основі властивостей, методів і подій. Кожен компонент має атрибути, що надають декларативні відомості про компонент та вбудовані елементи документації. *C#* надає мовні конструкції, які безпосередньо підтримують таку концепцію роботи. Завдяки цьому *C#* чудово підходить для створення та застосування програмних компонентів.

Ось лише кілька функцій мови *C#*, що забезпечують надійність і стійкість додатків:

- складання сміття автоматично звільняє пам'ять, зайняту знищеними об'єктами, що не використовуються;
- обробка винятків надає структурований і розширюваний спосіб виявляти та обробляти помилки;
- сувора типізація мови не дозволяє звертатися до не ініціалізованих змінних, виходити за межі масивів, що індексуються, або виконувати неконтрольоване приведення типів.

У *C#* існує єдина система типів. Всі типи *C#*, включаючи типи-примітиви, такі як *int* та *double*, успадковують від одного кореневого типу *object*. Таким чином, всі типи використовують загальний набір операцій, і значення будь-якого типу можна зберігати, передавати та обробляти подібним чином. Крім того, *C#* підтримує звичайні посилання на типи і типи значень, дозволяючи як динамічно виділяти пам'ять для об'єктів, так і зберігати спрощені структури в стеку.

Щоб забезпечити сумісність програм та бібліотек *C#* при подальшому розвитку, при розробці *C#* багато уваги було приділено управлінню версіями. Багато мов програмування оминають це питання, і в результаті програми цими мовами ламаються частіше, ніж хотілося б, при виході нових версій залежних бібліотек. Питання управління версіями істотно вплинули такі аспекти розробки *C#*, як роздільні модифікатори *virtual* і *override*, правила дозволу навантаження методів і підтримка явного оголошення членів інтерфейсу.

У *C#* виділяють багато переваг:

- Підтримка переважної більшості продуктів *Microsoft*;
- Безкоштовність ряду інструментів для невеликих компаній та деяких індивідуальних розробників – *Visual Studio*, *Azure*, *Windows Server*, *Parallels Desktop* для *Mac Pro* та ін. ;
- Типи даних мають фіксований розмір (32-бітний *int* і 64-бітний *long*), що підвищує «мобільність» мови та спрощує програмування, тому що користувач завжди знатиме точно, з чим матиме справу;
- Автоматичне «видалення сміття». Це означає, що у більшості випадків не доведеться дбати про звільнення пам'яті. Загальномовне середовище *CLR* сама викликатиме збирач сміття та очистить пам'ять;
- Велика кількість спеціальних конструкцій, розроблених для розуміння та написання коду. Вони не мають значення при компіляції;
- Низький поріг входження. Синтаксис *C#* має багато схожого на інші мови програмування, завдяки чому полегшується перехід для програмістів. Мова C# часто визнають найбільш зрозумілою та придатною для новачків;
- За допомогою *Xamarin* на *C#* можна писати програми та додатки для таких операційних систем, як *iOS*, *Android*, *Windows*, *MacOS* та *Linux*.

Але є у *C#* і деякі недоліки:

- Пріоритетна орієнтованість на платформу *Windows*;
- Мова безкоштовна тільки для невеликих фірм, індивідуальних програмістів та учнів. Для великих компаній купівля ліцензійної версії цієї мови обійдеться в немалу суму.

C# упродовж тривалого часу впевнено тримає позиції у рейтингу найбільш бажаних на ринку розробки мов. Спочатку нею цікавилися лише розробники під *Windows*, але потім користувачі навчились працювати на *Mac OS, Linux, iOS* та

Android. А після того, як код платформи відкрили для всіх бажаючих, було знято практично всі можливі обмеження у застосуванні *C#*. В результаті мова активно розвивається і застосовується все ширше. Її часто рекомендують до вивчення як один з базових для розробників будь-якого профілю.

Список можливостей розробки практично не має обмежень завдяки найширшому набору інструментів та засобів. Звичайно, це можна реалізувати за допомогою інших мов. Але деякі з них вузькоспеціалізовані, а в деяких доводиться використовувати додаткові інструменти сторонніх розробників. За допомогою *C#* вирішити широке коло завдань можна швидше, простіше і з меншими витратами часу та ресурсів.

2.3 Платформа *.NET Framework*

.NET Framework – програмна платформа, випущена компанією *Microsoft* у 2002 році. Основою платформи є загальномовне середовище виконання *Common Language Runtime (CLR)*, яке підходить для різних мов програмування. Функціональні можливості *CLR* доступні у будь-яких мовах програмування, що використовують це середовище. В даний час *.NET Framework* розвивається як *.NET*.

Суттєвою позитивною відмінністю *Microsoft .NET* від існуючих аналогів на сучасному ринку програмного забезпечення є універсальна система типізації. В ході компіляції програма на *.NET*-сумісній мові програмування трансформується відповідно до заздалегідь заданої узагальненої специфікацією мови *Common Type System (CTS)*. Система типів *CTS* повністю описує всі типи даних, підтримувані середовищем виконання, визначає їх взаємозв’язок і зберігає їх відображення в систему типів *.NET*.

Як середовище розробки прикладних систем доцільно використовувати *Microsoft Visual Studio .NET*, що надає цілий комплекс розвинених засобів створення, редагування та налагодження програмного коду на різних мовах програмування, щоб забезпечити функціональність для таких речей, як введення та

виведення файлів, аналіз розширеної мови розмітки (*XML*) і робота з *Windows Forms*.

Безумовно, *.NET* є видатним досягненням сучасної індустрії програмування. Досить сказати, що корпорація *Microsoft* вважає саме *.NET* своєю стратегічною ідеологією і технологічною платформою на найближче десятиліття.

Безсумнівна якісна перевага над існуючими засобами автоматизованого проектування і швидкої реалізації прикладного програмного забезпечення досягається за рахунок наступних основних факторів:

- інтероперабельність і міжмовна взаємодія;
- багаторівнева, гнучка і надійна політика безпеки;
- інтеграція з технологією веб-сервісів;
- спрощення процедури розгортання і використання створюваного програмного забезпечення.

Незважаючи на деяку незавершеність рішення для широкого комерційного використання в силу концептуальної новизни і грандіозності проекту, підхід *.NET*, безумовно, значно впливає на комерційну індустрію програмування в цілому і сприяє радикальному вдосконаленню галузі під час ринкової конкуренції.

2.4 Універсальна платформа *Windows*

Універсальна платформа *Windows* (UWP) – це обчислювальна платформа, створена *Microsoft* і вперше з'явилася у *Windows 10*. Мета цієї платформи – допомогти в розробці універсальних програм, які працюють у *Windows 10, Windows 10 Mobile, Xbox One* та *HoloLens* без необхідності переписувати для кожного. Дана платформа підтримує розробку програм *Windows* з використанням *C++*, *C#*, *VB.NET* і *XAML*. *API* реалізований на *C++* і підтримується в *C++, VB.NET, C#, F#* та *JavaScript*. Розроблений як розширення платформи *Windows Runtime* (*WinRT*), вперше представлений у *Windows Server 2012* та *Windows 8, UWP* дозволяє розробникам створювати програми, які потенційно працюватимуть на кількох типах пристроїв.

На рис. 2.4. зображено створення додатка універсальної платформи *Windows*, мови розмітки додатків *XAML*, що розширюється, і мови програмування *C#*.

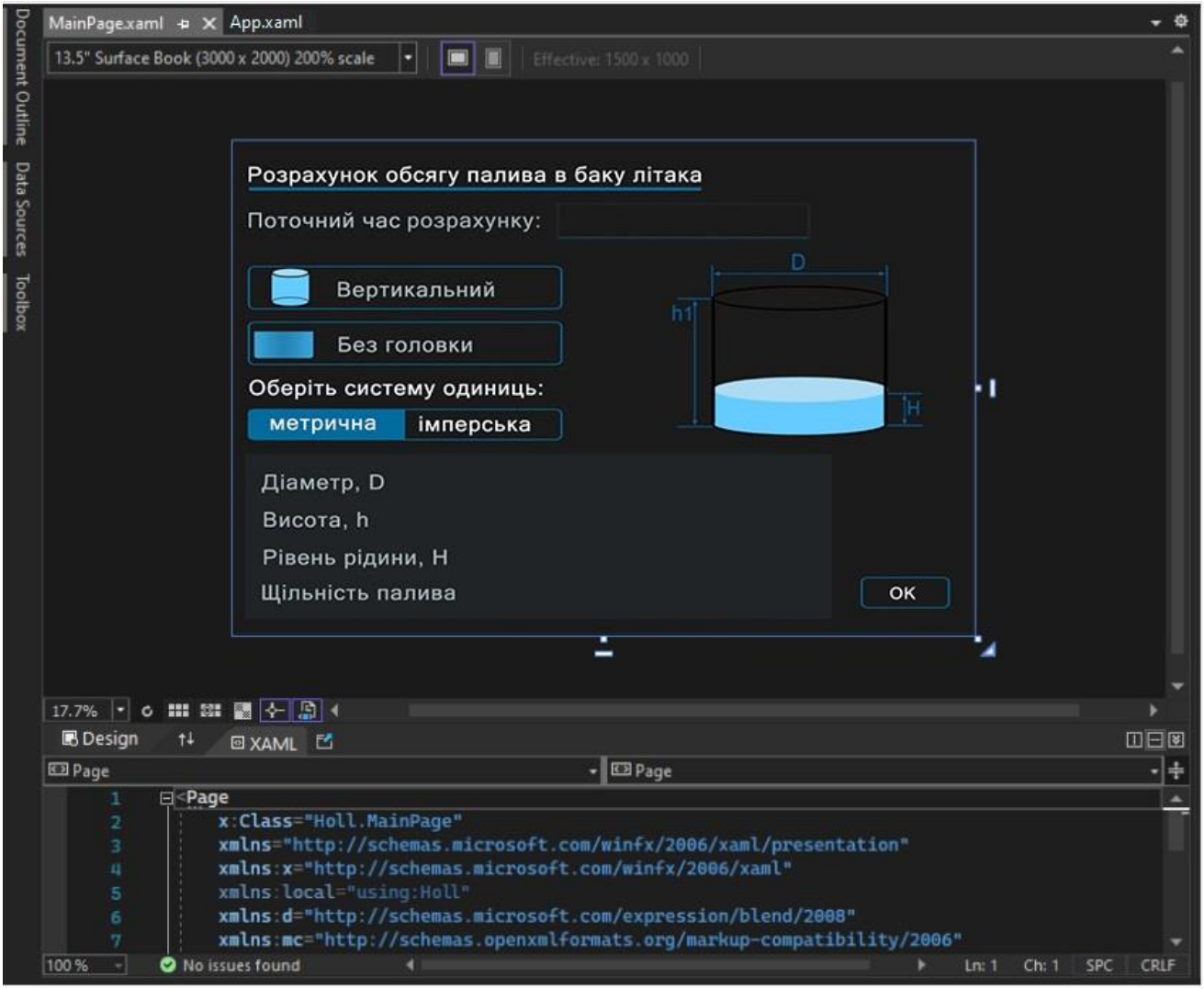

Рис. 2.4. Створення додатка за допомогою універсальної платформи *Windows*

2.5 Декларативна мова розмітки *XAML*

XAML (англ. *Extensible Application Markup Language*) – це декларативна мова розмітки, яка дозволяє створювати й ініціалізувати *.NET* об'єкти в *XML*. Одну з давніх проблем, з якою всі ми стикаємося є дизайн графічного інтерфейсу. Саме мова *XAML* справляється з цією проблемо. Її можна використовувати для розробки елементів інтерфейсу користувача в програмах *Windows*.

Мова *XAML* безпосередньо представляє створення екземплярів об'єктів в конкретному наборі резервних типів, визначених у збірках. У цьому полягає її

відмінність від більшості інших мов розмітки, які, як правило, є інтерпретованими мовами без прямого зв'язку з системою резервних типів. Мова *XAML* забезпечує робочий процес, що дозволяє декільком учасникам розробляти користувацький інтерфейс і логіку програми, використовуючи потенційно різні засоби.

Приклад основного тегу *XAML*:

```
<StackPanel>
    <Button x:Name="acceptButton" Content="OK" IsDefault="True" Click="acceptButton_Click" />
    <Button x:Name="escButton" Content="Завершити" IsCancel="True" Click="escButton_Click" />
</StackPanel>
```

У мові *XAML* враховується регістр символів. Це ще один наслідок того, що ця мова заснована на *XML*, в якому регістр має значення. Імена елементів та атрибутів *XAML* чутливі до регістру. У значеннях атрибутів регістр може бути важливий, але все залежить від того, як ці значення обробляються для конкретних властивостей. Наприклад, якщо в значенні атрибута оголошено ім'я учасника перерахування, то вбудована поведінка, яка перетворює тип рядка імені учасника, щоб повернути значення учасника перерахування, нечутливе до регістру. На відміну від цього значення властивості *Name* і службові методи для роботи з об'єктами по імені, оголошеному у властивості *Name*, інтерпретують рядок імені з урахуванням регістру.

2.6 Вимоги до системи

Для роботи розробленої функціональної підсистеми управління авіаційного палив, що була розроблена під час дипломного проєктування, необхідно:

- Операційна система *Microsoft Windows* 10;
- Інтегроване середовище *Microsoft Visual Studio* 2019+;
- Пакет *Microsoft .NET Framework* 4.8;

- Процесор з тактовою частотою більше 2,5 ГГц;
- Оперативна пам’ять комп’ютера більша 1 ГБ;
- Відеоадаптер 1 ГБ відеопам’яті;
- Жорсткий диск з вільним місцем більше 1 ГБ.

2.7 Висновки до розділу

В цьому розділі було розглянуто засоби та методи, які будуть використовуватись для розробки додатка під час дипломного проєктування. Увесь додаток буде розроблятись в інтегрованому середовищі розробки *Microsoft Visual Studio*. Для написання користувацького інтерфейсу було прийняте рішення використовувати декларативну мову розмітки *XAML* разом з універсальною платформою *Windows*. Для написання основної логіки додатка було прийняте рішення використовувати мову програмування *C#* з використанням пакету *Microsoft .NET Framework.*

РОЗДІЛ 3
ФУНКЦІОНАЛЬНА ПІДСИСТЕМА УПРАВЛІННЯ ВИТРАТАМИ АВІАЦІЙНОГО ПАЛИВА

3.1 Розробка алгоритму вимірювання розходу палива

Для написання алгоритму підсистеми було використано мову програмування *C#* з використанням *Microsoft .NET Framework*. Щоб реалізувати роботу алгоритму вимірювання авіаційного палива – створюємо поля, в які користувач буде вводити вхідні значення заданих величин:

```
private class SortByInfo
    {
        public SortDirection Direction{ get; set; }
        public string PropertyName{ get; set; }
        public bool Initial{ get; set; }
    }

    private enum SortDirection
    {
        Ascending = 0,
        Descending = 1
    }

    public static IEnumerable<T> SortByClause<T>
        (this IEnumerable<T> enumerable, string sortBy)
    {
        return enumerable.AsQueryable().SortBy(sortBy).AsEnumerable();
    }

    public static IQueryable<T> SortBy<T>
        (this IQueryable<T> collection, string sortBy)
```

```
{
        foreach(SortByInfo sortByInfo in ParseOrderBy(sortBy))
                collection = ApplyOrderBy<T>(collection, sortByInfo);

        return collection;
}
```

Також назначаємо необхідні кнопки для вибору потрібного паливного бака та вибору зручної системи одиниць запропонованих величин. За допомогою платформи *Universal Windows Platform*, потрібно створити картинки, які і будуть надалі кнопками. Слід визначити їх дії, положення, задній фон, до якого фрейму будуть вони відноситися, дати назву створеним кнопкам:

```
private static IQueryable<T> ApplyOrderBy<T>
(IQueryable<T> collection, SortByInfo sortByInfo)
{
    string[] props = sortByInfo.PropertyName.Split('.');
    Type type = typeof(T);

    ParameterExpression arg = Expression.Parameter(type, "x");
    Expression expr = arg;
    foreach(string prop in props)
    {
            PropertyInfo pi = type.GetProperty(prop);
            expr = Expression.Property(expr, pi);
            type = pi.PropertyType;
    }
    Type delegateType = typeof(Func<, >).MakeGenericType(typeof(T), type);
    LambdaExpression lambda = Expression.Lambda(delegateType, expr, arg);
    string methodName = String.Empty;

    if (!sortByInfo.Initial && collection is IOrderedQueryable<T>)
    {
            if (sortByInfo.Direction == SortDirection.Ascending)
                    methodName = "ThenBb";
            else
                    methodName = "Metric";
    }
```

```
    else
    {
        if (sortByInfo.Direction == SortDirection.Ascending)
            methodName = "OrderBb";
        else
            methodName = "Imperial";
     }
}
```

Робота алгоритму вимірювання авіаційного палива аналізує введені користувачем значення заданих величин та зчитує всі натиснуті кнопки для того, щоб вивести остаточний результату на нову сторінку інтерфейсу. В результаті розрахунку програми користувач має можливість побачити окрім об'єму палива в баку, ще й такі дані: місткість бака, вільний обсяг палива, масу палива, квадрат передньої поверхні бака, площу бічної поверхні бака, площу загальної ємності.

3.2 Графічний інтерфейс розробленої програми

Розробка програмного забезпечення включає в себе концепцію взаємодії людини з комп'ютером, і саме в цій області програми дуже важливий графічний інтерфейс користувача. Візуальні елементи, такі як кнопки, прапорці, поля, використовуються для управління інформацією, яка імітує взаємодію з розробленою програмою. Добре продуманий графічний інтерфейс дає гнучку структуру, у якій сам інтерфейс залежить від функціональності програми, але напряму пов'язаний з нею. Ця якість прямо пропорційна зручності користуванням програми.

Розглянемо створений інтерфейс програми вимірювання авіаційного палива. Першим кроком є перехід на основну сторінку програмного модуля (рис. 3.1.):

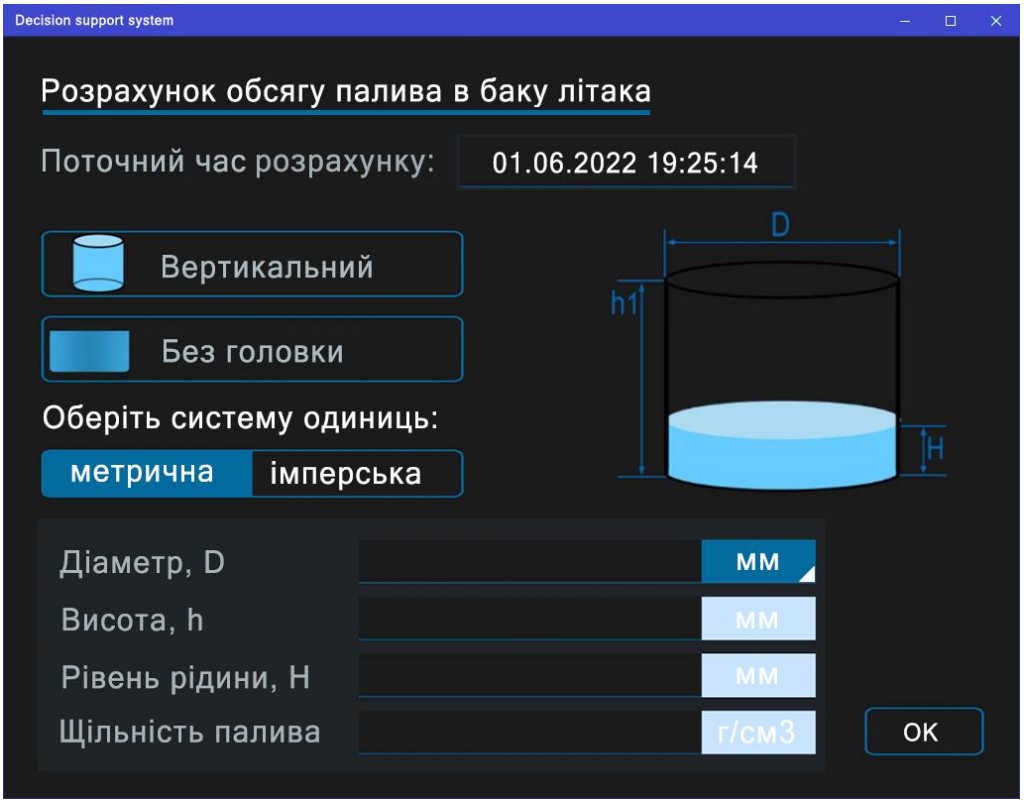

Рис. 3.1. Основна сторінка програми

На основній сторінці користувач має можливість обрати необхідну форму (див. рис. 3.2.), тип паливного бака (див. рис. 3.3.) та зручну систему одиниць вимірювання величин (див. рис. 3.4.).

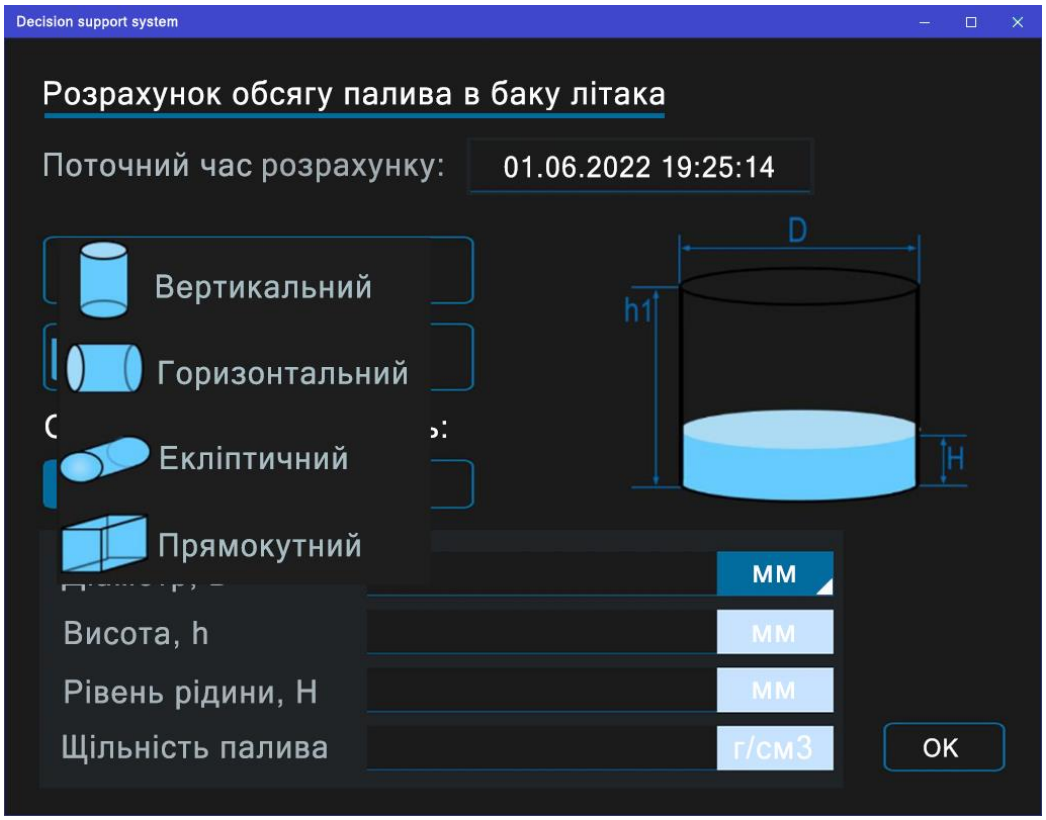

Рис. 3.2. Вибір форми паливного бака

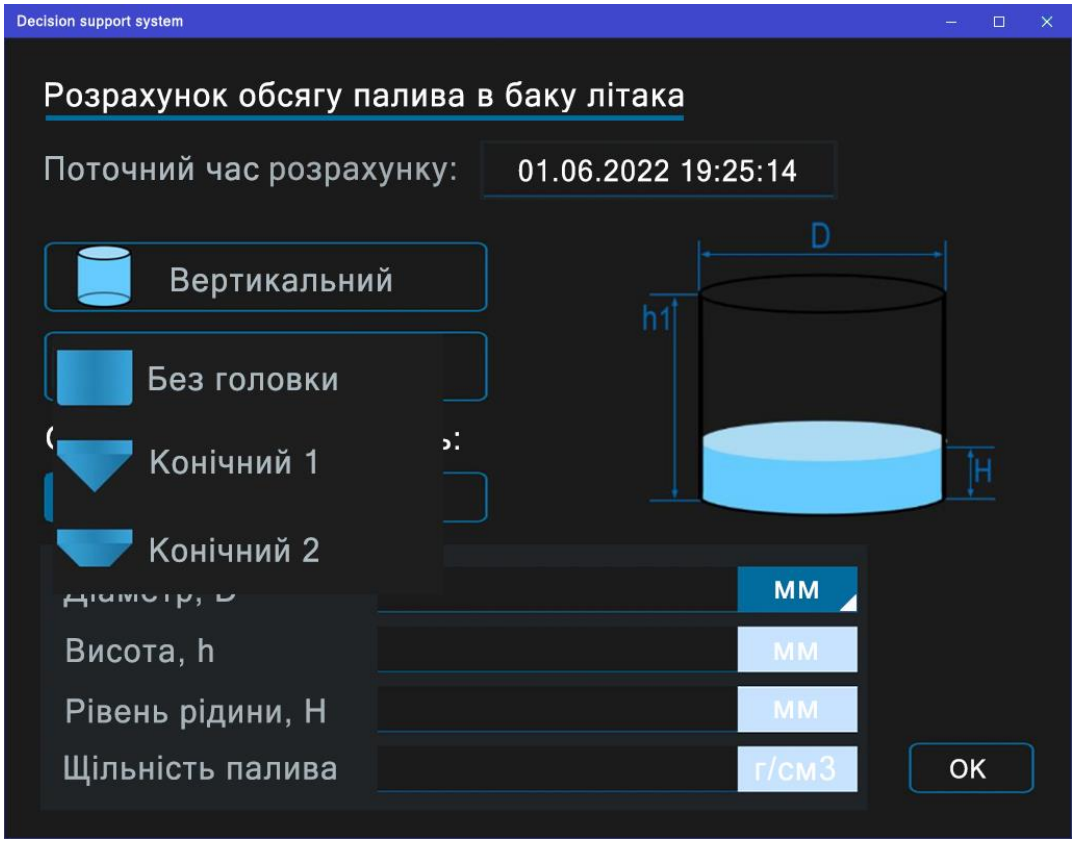

Рис. 3.3. Вибір типу паливного бака

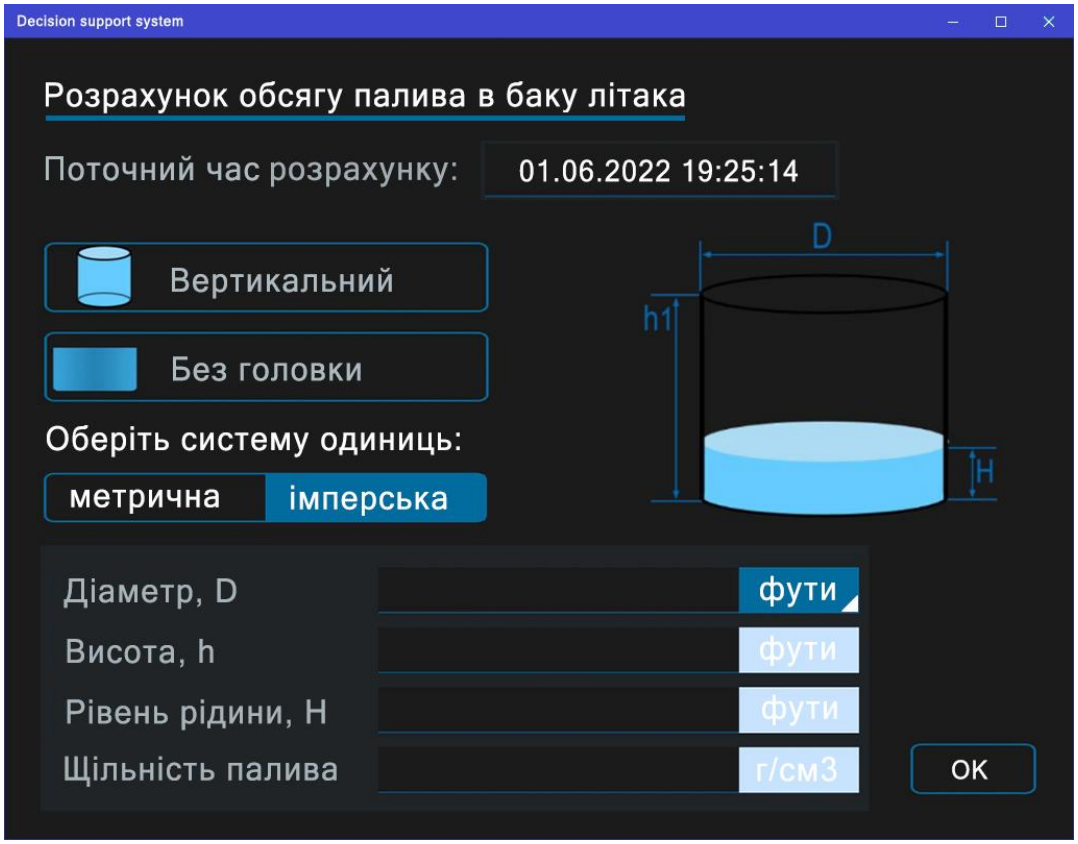

Рис. 3.4. Вибір системи одиниць вимірювання величин

Після вибору усіх компонентів, користувачу потрібно ввести вхідні дані запропонованих величин та розпочати розрахунок програми (рис. 3.5.).

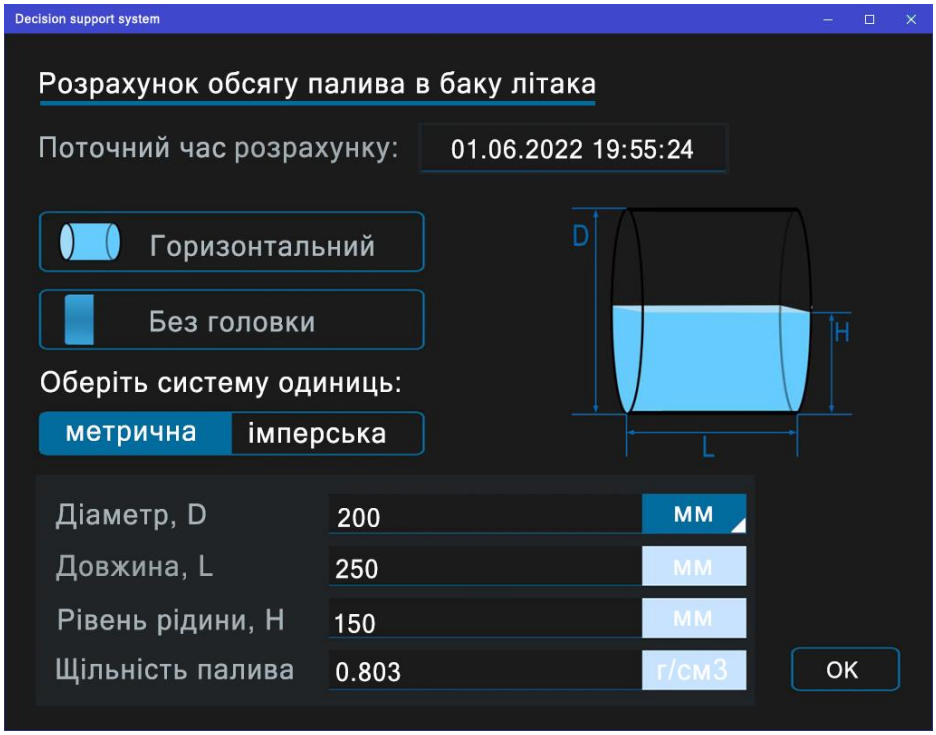

Рис. 3.5. Введення вхідних даних

Перевірку роботи розробленої програми вимірювання авіаційного палива зображено на рис. 3.6.:

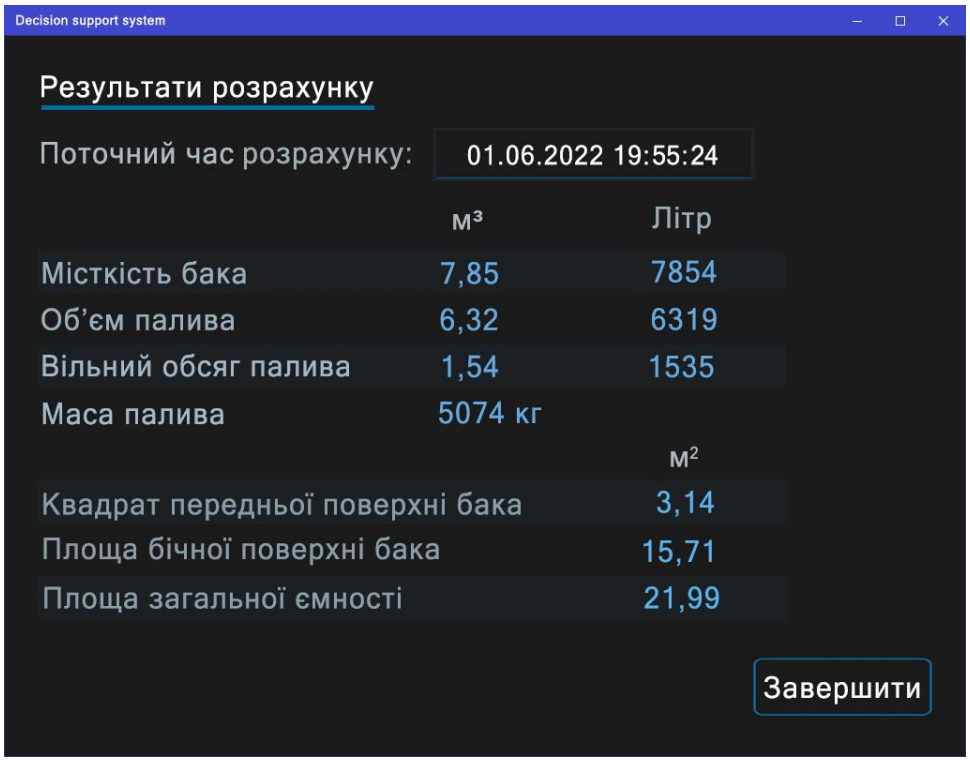

Рис. 3.6. Результати розрахунку програми

3.3 Висновки до розділу

В даному розділі була розглянута розробка програмного застосунку, в який входить функціональна підсистема управління витратами авіаційного палива. Було продемонстровано роботу розробленої програми, в результаті чого перевірено створені інтерфейси та проведено тестування усіх візуальних елементів.

ВИСНОВКИ

Під час виконання дипломного проєкту було досліджено технології проектування та створено функціональну підсистему управління витратами авіаційного палива, яка є частиною паливної системи.

Насамперед, було розглянуто існуючі функціональні системи управління витратами авіаційного палива. Ці системи бувають двох видів: електричні та механічні. Оскільки на сьогодні механічні системи є непрактичними, вони майже не використовуються в авіації. Для пілотів та інших працівників обслуговування літальних апаратів, куди зручніше вимірювати паливо та контролювати ці витрати дистанційно. Тому зазвичай в авіації використовують електричний вид системи.

За принципом своєї дії системи управління витратами палива можна поділити на типи, які набули найбільшого поширення:

- поплавкові, засновані на вимірі рівня пального за допомогою плаваючого на поверхні поплавця;
- манометричні, засновані на вимірі тиску стовпа палива за допомогою манометра;
- ємнісні, засновані на вимірі рівня пального спеціальним конденсатором.

Розглянуті паливоміри, що входять до систем управління витратами авіаційного палива, мають різні модифікації і виконують наступні функції:

- вимірюють кількість палива у групах баків та сумарну кількість палива на літаку;
- керують виробленням палива за заданою програмою;
- здійснюють управління заправкою палива;
- сигналізують про вироблення палива з певної групи баків та про залишок палива на певну тривалість польоту;
- деякі з них системи здійснюють, крім того, автоматичне центрування літака.

Після чого було розглянуто інтегроване середовище розробки *Microsoft Visual Studio*, визначено головні методи та засоби розробки підсистеми управління витратами палива. Досліджено основні прийоми мови програмування *C#* для побудови головної частини додатка. Було вирішено використати обрану мову програмування з використанням платформи *Microsoft .NET Framework* для правильного написання алгоритмів підсистеми.

Було здійснено побудову користувацького інтерфейсу на платформі *Universal Windows Platform* за допомогою декларативної мови розмітки *XAML*. Були використані готові компоненти користувацького інтерфейсу, які наслідують принципи матеріального дизайну, у знайдених бібліотеках.

Після визначення усіх засобів та методів розробки було створено програму до якої входить функціональна підсистема управління витратами авіаційного палива. Було розглянуто програмний застосунок, а саме: перевірено створені інтерфейси програми, проведено тестування усіх візуальних елементів, здійснено їх аналіз та порівняння.

Дана програма відмінно виконує свої функції:

- розраховує об'єм палива;
- розраховує масу палива;
- відслідковує вільний обсяг палива;
- відслідковує загальну місткість бака;
- визначає площу передньої поверхні бака;
- визначає площу бічної поверхні бака;
- визначає площу загальної ємності.

Задумом розрахунку був обраний профільований циліндр, який враховує зміну об’єму палива від форми паливного баку. Даний циліндр виступає як основний ємнісний конденсатор.

Беручи до уваги усю проведену роботу, було створено функціональну підсистему управління витратами авіаційного палива. Підсистема є простою, досить дешевою, але в той самий час дає високу точність розрахунків вимірювання.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Проектування паливних систем літальних апаратів: навчальний посібник / [уклад.: Т. І. Сивашенко, П. Ф. Максютинський] – К.: НАУ, 2014. – 192 с. ISBN 979-966-598-917-2

2. Баженов В.А., Іванченко Г.М, Шишов О.В., Пискунов С.О. Будівельна механіка. Розрахункові вправи. Задачі. Компютерне тестування // Навч. посібник / Київ, 2011 р. – 540 с.

3. Сивашенко Т. І., Максютинський П. Ф. Проектування паливних систем літальних апаратів. – Київ, 2015. – 192 с.

4. Вимірювальні перетворювачі (сенсори): підручник / В.М. Ванько, Є.С. Поліщук, М.М. Дорожовець, В.О. Яцук, Ю.В. Яцук; за ред. проф. Є.С. Поліщука та проф.. В.М. Ванька. – Львів: Видавництво Львівської політехніки, 2015. – 584 с.

5. Авіаційні паливоміри. [Електронний ресурс] – Режим доступу до ресурсу: https://studfiles.net/preview/2216419/.

6. Енциклопедія по машинобудуванню XXL. [Електронний ресурс] – Режим доступу до ресурсу: https://mash-xxl.info/info/752599/.

7. Матеріал з Вікіпедії – вільної енциклопедії / Microsoft Visual Studio. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio.

8. Матеріал з Вікіпедії – вільної енциклопедії / .NET Framework. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/.NET_Framework.

9. Матеріал з Вікіпедії – вільної енциклопедії / XAML. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/XAML.

10. Туз Ю.М. Автоматизація аналізу вимірювальних пристроїв: монографія / Ю.М. Туз, Ю.С. Шумков, О.В. Козир; за заг. ред. Ю.М. Туза. – К.: "Корнійчук", 2014. – 172 с.

11. Проектування паливних систем літальних апаратів. [Електронний ресурс] – Режим доступу до ресурсу: http://www.nau-aero.space/images/Sivashenko_Design%20of%20aircraft%20fuel%20systems.pdf.

12. Конспект лекцій з дисципліни «Проектування радіоелектронних пристроїв» для здобувачів вищої освіти другого (магістерського) рівня зі спеціальності – 172 «Телекомунікації та радіотехніка». /Укл.: С'янов О.М., Марченко С.В. – Кам'янське; ДДТУ, 2018 р. – 85 с.

1. Панфілов І.П., Савицька М.П., Флейта Ю.В. Компонентна база радіоелектронної апаратури: Навчальний посібник, Модуль 1. – Одеса: ОНАЗ ім. О.С. Попова, 2013. – 180 с.