ЗМІСТ

ВСТУП

Сучасне та ефективне управління міським господарством та розвитком територій полягає у створенні та впровадженні службових програмних засобів, переході з паперового на електронний документообіг, а також перекваліфікації працівників.

Необхідність впровадження процесів автоматизації у систему контролю управління водними ресурсами для підвищення ефективності використання водопостачання.

Потреба у впровадженні новітніх технологій та удосконаленні вже існуючих. Найкраще це реалізувати за допомогою створення сучасних програмних забезпечень, які повинні стати основою для формування інформаційної системи в управлінні міським господарством.

Метою курсової роботи є розробка програми розрахунку втрат води, при пошкодженнях трубопроводу. Дослідження існуючого стану використання інформаційних технологій на прикладі приватного акціонерного товариства «Київводоканал» та закріплення набутих знань і практичних навичок з дисципліни «Алгоритмізація та програмування».

Завдання курсової роботи:

- розглянути сучасний стан інформаційного забезпечення розрахунку втрат води при пошкодженнях трубопроводу в Пр АТ АК «Київводоканал»;
- розробити алгоритм та програму розрахунку втрат води при пошкодженнях трубопроводу;
- дослідити ефективність та протестувати розроблену програму.

Об’єктом дослідження є методи розрахунку втрат води в Пр АТ АК «Київводоканал».

Предметом дослідження є Інформаційне забезпечення в управлінні водними ресурсами.

Інформаційною базою курсової роботи є дослідження науковців у сфері алгоритмізації та програмування та інформаційні системи, які

використовуються для управління водними ресурсами.

Під час виконання курсової роботи для написання коду програми було використано такі середовища розробки: DEV C++, CodeBlocks. Для пошуку інформації в мережі інтернет – веб-браузер Google Chrome, для оформлення текстової частини пояснювальної записки - компоненти пакету Microsoft Office, зокрема, текстовий редактор Word.

Курсова робота містить 50 сторінок, 4 таблиці, 21 рисунків, список використаної літератури з 12 найменувань, 2 додатки.

РОЗДІЛ 1. ДОЦІЛЬНІСТЬ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ПРОЕКТУВАННЯ

1.1.Завдання та суть розробки

Завданням курсової роботи є створення програми розрахунку втрат води при пошкодженнях трубопроводу. На сьогодні, у департаментах Водоканалу втрати води при пошкодженнях трубопроводу рахуються вручну, за формулами. Тому й виникла необхідність написати програму розрахунку для автоматизації та оптимізації подальшої роботи.

Для реалізації завдання необхідно розробити послідовні алгоритми, які будуть послідовно виконувати розрахунки та реалізувати графічний інтерфейс для користувача[1].

Для цього використаємо мову програмування «C++», середовища розроблення DEV C++, CodeBlocks, та модуль програмування Windows – Windows API, для створення графічного інтерфейсу в даній операційній системі.

WinAPI – це основний набір програмних інтерфейсів (API) Microsoft доступних в операційних системах Windows (раніше Win32 API). Це інтерфейс прикладного програмування, який використовується для створення програм Windows[2]. На рисунку 1.1. вказано які програмні компоненти можна створювати за допомогою WinAPI.

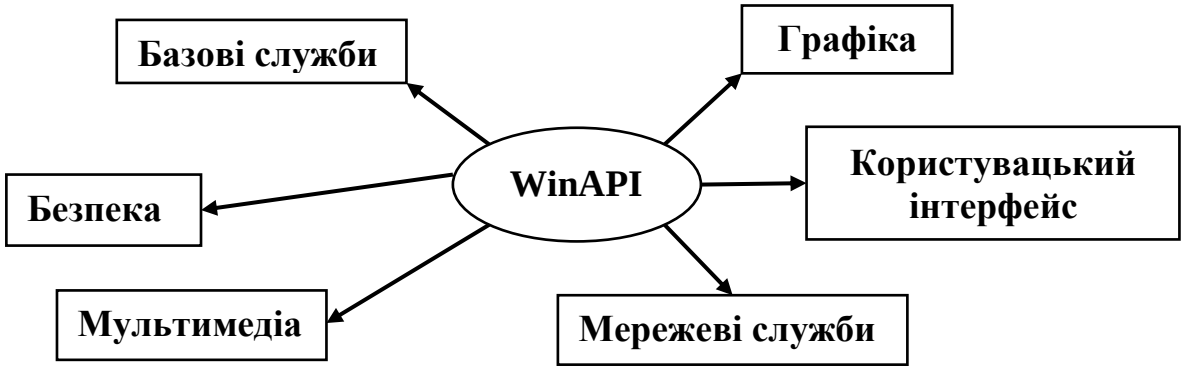


Рисунок 1.1 – Програмні компоненти WinAPI

Базові служби забезпечують доступ до основних ресурсів. До них відносяться функції, файлові системи, пристрої, процеси, потоки, реєстр та обробка помилок. Область безпеки надає інтерфейси, об’єкти і інші елементи програмування для аутентифікації, авторизації, криптографії і інших пов’язаних з безпекою завдань.

Підсистема графіки забезпечує функціональність виводу графічного складу на монітори, принтори та інші пристрої. Користувацький інтерфейс забезпечує функціональність для створення вікон та елементів управління. Компонент мультимедіа надає інструменти для роботи з відео та звуковими пристроями. Мережеві служби надають доступ до мережевих можливостей Windows [3].

В дану платформу входить SDK (Software Development Kit), що включає файли з заголовків, бібліотек, документації та інструментів, що використовуються для розробки програмних забезпечень. WinAPI створений для мов програмування C і C++. Це прямий спосіб створення програм в операційній системі.

1.2.Аналіз сучасного стану застосування програм аналогів

На сьогодні, є програми загальних розрахунків всієї мережі водопостачання, а програм які розраховують втрати води певної ділянки водопроводу – немає. Інженери переважно прораховують вручну за формулою:

$$Q = 2{,}65*F*t*\sqrt{H} \qquad \text{…….……(1.1)}$$

$$F = \pi R2 \qquad (1.2)$$

де t – час, H- тиск води, R – радіус пошкодження, F – площа пошкодження.

В Додатку Б вказана порівняльна характеристика програм аналогів, які розраховують загальні втрати води, гідравлічні розрахунки мережі водопостачання підбір насосних агрегатів та ін.

1.3.Обґрунтування вибору мови та середовища програмування

Мова програмування С з’явилася в 1972 році і використовується донині для розробки операційних систем, драйверів, ігор. Певна модифікація цієї мови використовується для розробки програм для мобільних пристроїв. Мова С ++ була розроблена на основі С і з'явилася на початку 1980-х років. Вона досить універсальна і використовується для створення мікроконтролерів, роботів, ігор, систем моделювання, обробки статистики і в нейронних мережах. Загалом, не існує області програмування, де С ++ можна було б застосувати [4].

З мови С легко перейти на інші сучасні мови. Java, C #, Objective C, Java Script - все це «С-подібні» мови, синтаксис яких буде інтуїтивно зрозумілий кожному програмісту, який знає С / С ++ [5].

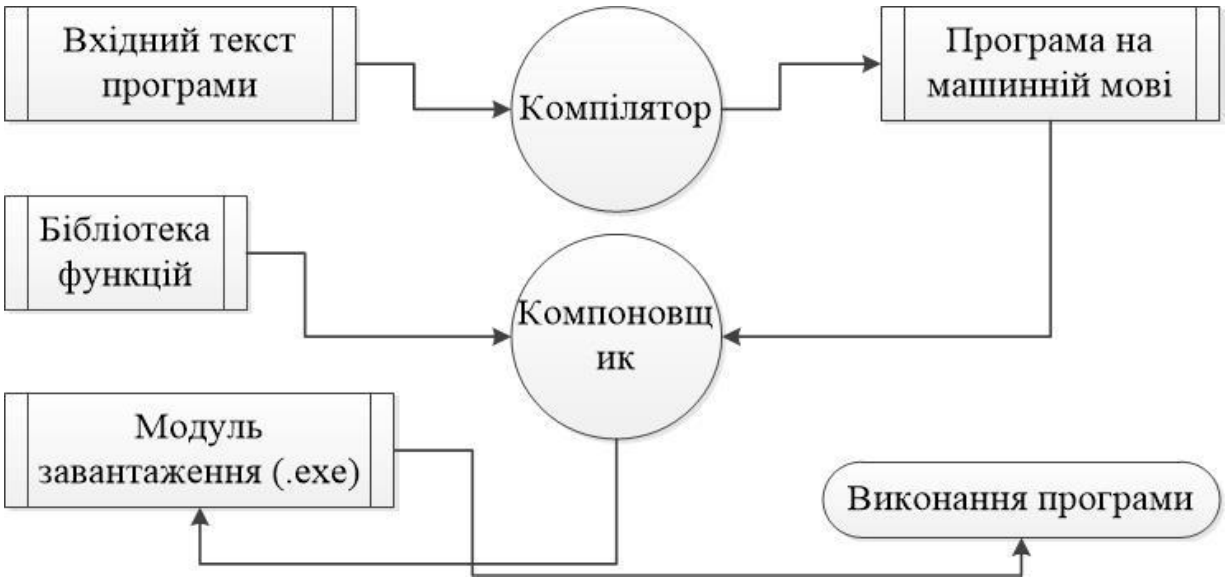


Рисунок 1.6 – Схема інтегрованого середовища розробки

Інтегроване середовище розробки (IDE – Integrated Development Environment), схематично вказане на рисунку 1.6 – це спеціальний додаток, який дозволяє спростити розробку програм на мові високого рівня. Воно зазвичай включає в себе [6]:

- текстовий редактор – щоб набрати текст програми, зберегти його на диску, редагувати. Такий редактор зазвичай ще й різними кольорами

виділяє елементи коду, підказує можливі помилки синтаксису. А іноді і створює фрагменти коду автоматично.

- компілятор і компонувальник – щоб перевести програму в машинний код;
- засоби налагодження і запуску програм – щоб забезпечити зручності пошуку помилок;
- стандартні бібліотеки, що містять багато разів використовувані елементи програм;
- довідкову систему і ін.

Для мови С існує досить багато різних середовищ розробки. В даній роботі використано DEV C++ та CodeBlocks (Рисунок 1.7 – 1.8). Дані системи є кросплатформні.

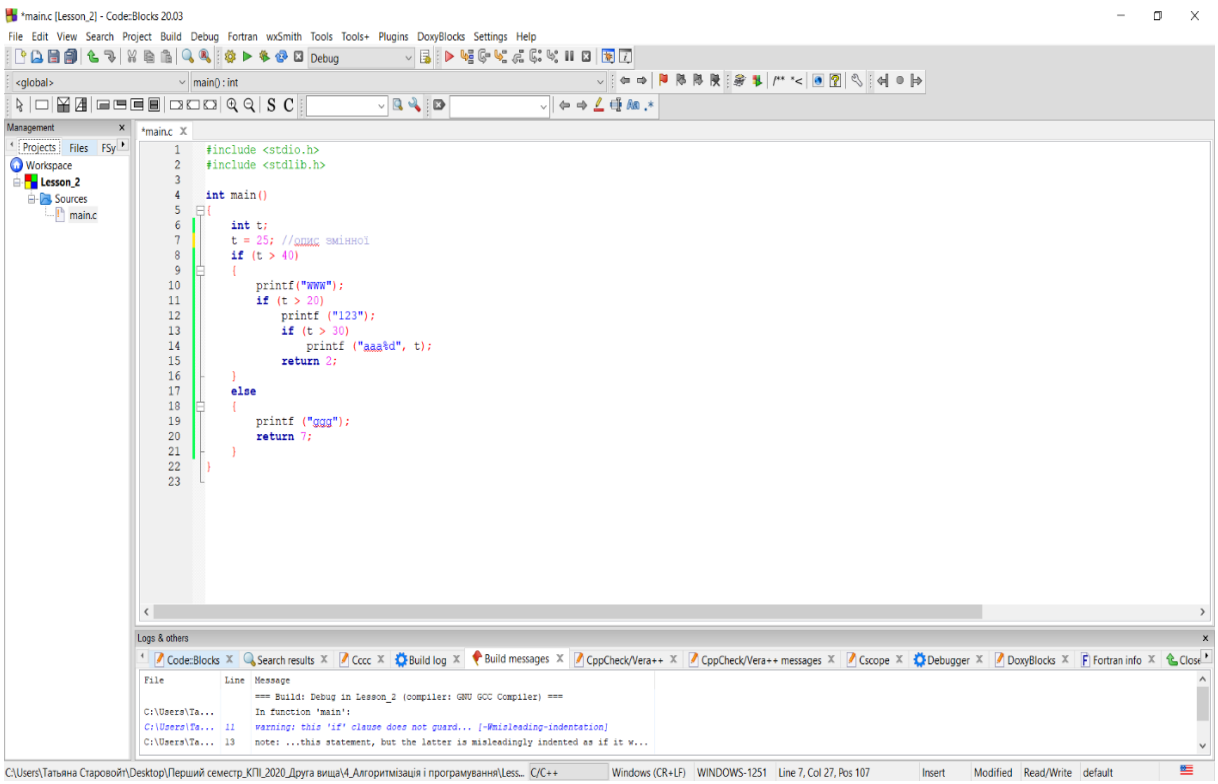


Рисунок 1.8 – Графічний інтерфейс середовища розробки Code::Blocks

В таблиці 1.2. детально описані середовища розробки для мови С/С++, та вказані їх переваги та недоліки. Інтерфейс компіляторів вказаний на рисунках: 1.7, 1.9, 1.10, 1.11, 1.12.

Таблиця 1.2

Порівняльна характеристика програм аналогів

№ П/П	Середовище розробки	Порівняльна характеристика середовищ розробки
1.	Eclipse	Одна з найбагатших функціоналом IDE з відкритим вихідним кодом. Спочатку вона головним чином використовувалася для розробки на Java, але зараз підтримує більшу різноманітність мов.
2.	NetBeans	Одна з кращих IDE для програмування на C і C ++. Вона має дружній до користувача інтерфейс, а також кілька приголомшливо корисних шаблонів проектів. Є функціонал drag-and-drop. Netbeans написана на Java, але надає повну підтримку і набір інструментів, необхідних для розробників, які пишуть на C і C ++.
3.	Visual Studio	Visual Studio Code від Microsoft це одна з найнадійніших і функціональних IDE, доступних для Windows, Linux і MacOS. Ця IDE базується на фреймворку Electron.
4.	Code::Blocks	Підтримує мало мов, але зате для них є однією з кращих IDE. Розробники, які пишуть на C і C ++, оцінять її налаштованість і гнучкість. Найкраще в Code::Blocks це доступність безлічі плагінів.
5.	DEV C++	Безкоштовне середовище розроблення для мов C/C++. Відрізняється невеликим розміром та найкращий варіант для початківців.

Переваги компіляторів:

Eclipse доступна для Windows, Linux і MacOS. Це середовище надає багато можливостей, таких як автоматичний аналіз коду, інтеграція git, статичний аналіз коду і т.д. Мови: C, C ++, C #, Java, JavaScript, Perl, PHP, Python, COBOL і т. д [7].

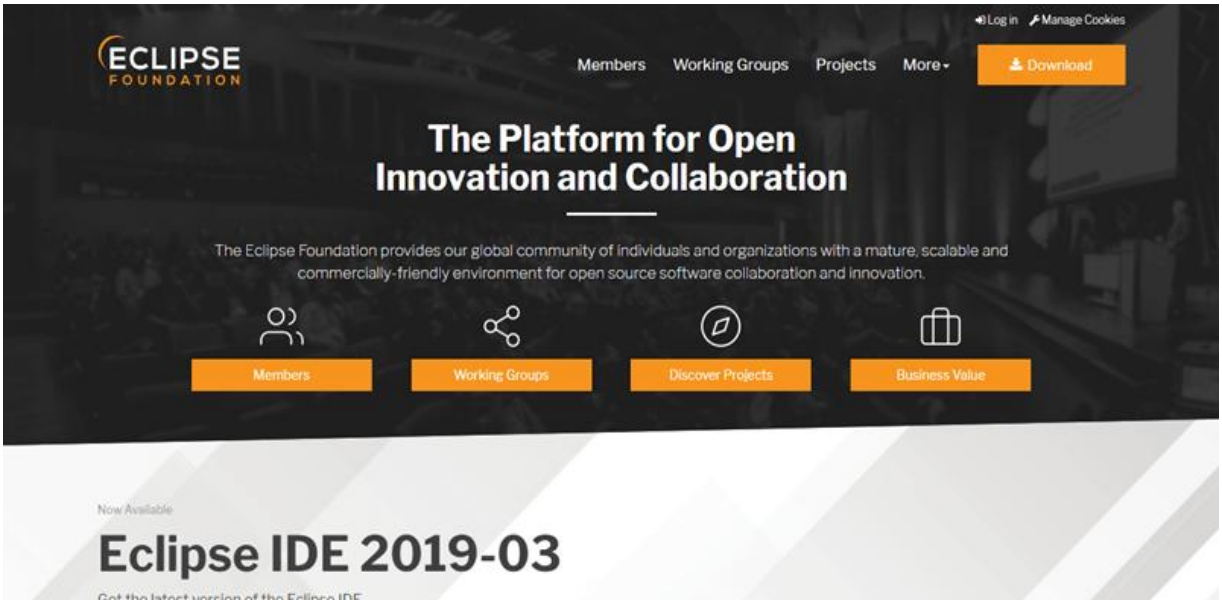


Рисунок 1.9 – Інтерфейс компілятора Eclipse

Найкраще в Netbeans це її прості й ефективні інструменти для управління проектами. Що поставляється функціонал можна розширити за допомогою різноманітних корисних плагінів. З NetBeans можливо моніторити розробку проекту віддалено [8].

Це середовище доступне для Windows, Mac OS X, Linux і Solaris.

Мови: C, C ++, Java, HTML, HTML 5 і інші.

Якщо говорити про функціонал, Visual Studio Code володіє всіма потрібними властивостями, такими як розумне доповнення коду, підсвічування синтаксису, рефакторинг коду, підтримка сніпетів, можливості налагодження, інтегрований контроль Git і т. Д.

Мови: C, C ++, C #, CSS, Go, HTML, Java, JavaScript, Python, PHP, TypeScript і т. д.

Функціонал цього середовища розробки можна як завгодно розширити, в тому числі за допомогою призначених для користувача плагінів.

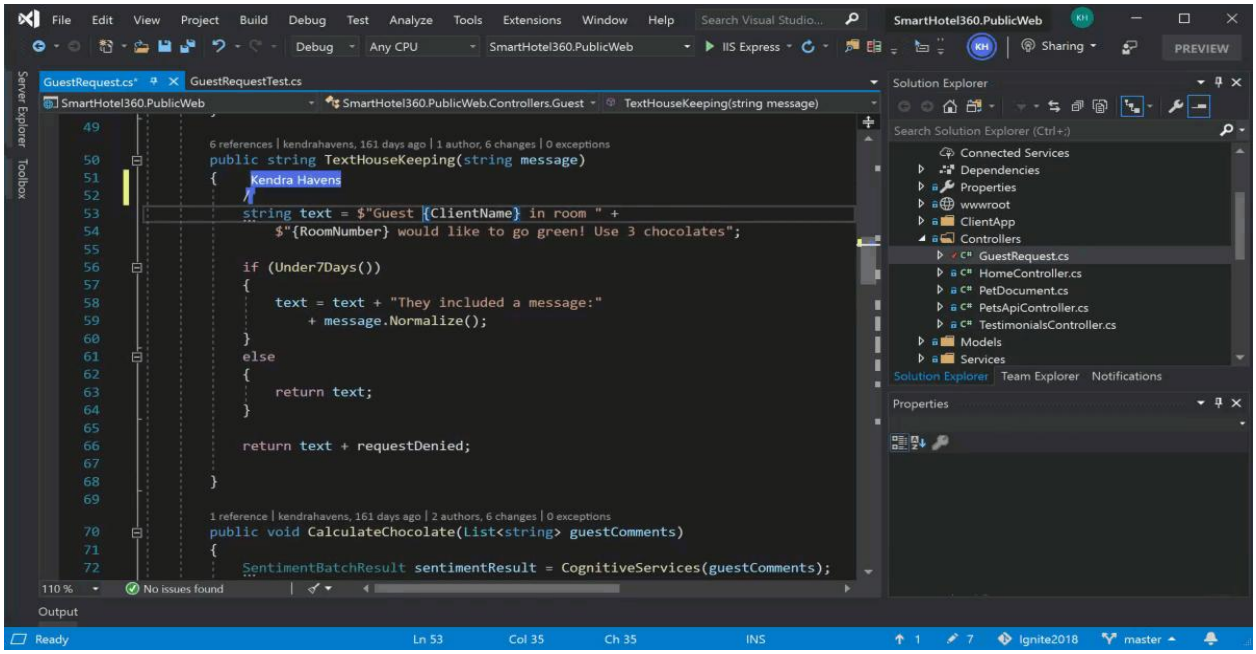


Рисунок 1.12 – Інтерфейс компілятора Visual Studio

Розумна підсвічування синтаксису, автодоповнення коду і повнофункціональний відладчик, наявні в Code::Blocks, зроблять розробку дійсно швидкою.

Code::Blocks доступна на всіх платформах, включаючи Windows, Linux і MacOS.

Мови: C, C ++ і Fortran.

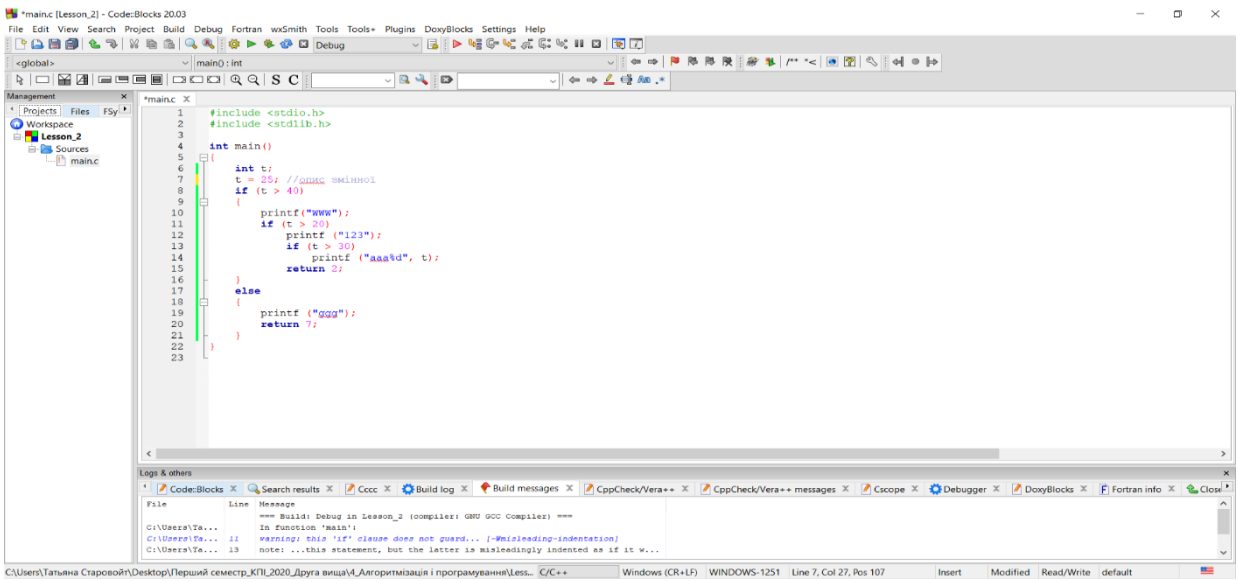


Рисунок 1.10 – Інтерфейс компілятора Code::Blocks

Atom це один з найпопулярніших і високо настроюються редакторів коду. Він розроблений Github і є прекрасним вибором як для маленьких, так і для великих проектів. Доступний для Windows, Linux і OS X, поставляється з дуже мінімалістичним призначеним для користувача інтерфейсом [10].

Ця IDE володіє всім необхідним функціоналом. У ній є менеджер пакетів, кілька панелей, функція пошуку і заміни тексту, підтримка палітри команд і т. д. Крім того, для Atom є безліч плагінів: з їх допомогою можна розширити функціонал цього редактора і зробити його ще більш корисним.

Мови: C / C ++, CSS, HTML, JavaScript, PHP, Python, Ruby і т. д.

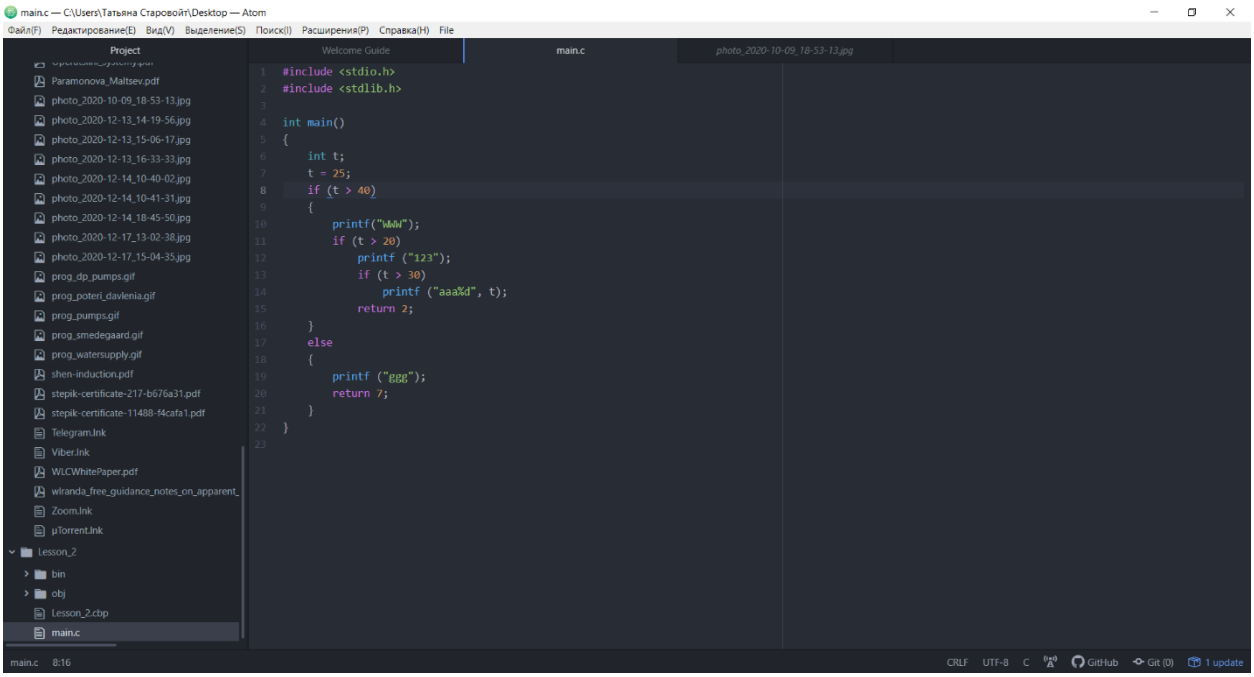


Рисунок 1.11 – Інтерфейс компілятора Atom

1.4. Постановка задачі проектування

Метою даної роботи є розробка програми розрахунку втрат води, при пошкодженнях трубопроводу. Для досягнення поставленої цілі необхідно:

1. розглянути і проаналізувати підходи до розв’язку задачі створення програми розрахунку втрат води;
2. обґрунтувати вибір технічних засобів для розв’язку завдань

проектування;

3. визначити та розглянути алгоритми, проаналізувати програми-аналоги, вибрати базовий варіант для розробки;

4. розробити структуру додатку та алгоритм його роботи;

5. розробити програмний засіб з розрахунку втрат води при пошкодженнях трубопроводу;

6. перевірити правильність та коректність роботи програми на контрольних прикладах.

РОЗДІЛ 2. МЕТОДИ ТА МАТЕМАТИЧНА ОСНОВА ПРОГРАМИ

2.1. Метод розв’язку задачі

Для виконання задачі використаємо ідею ООП, яка полягає в тому, що при створенні програмних структур імітується поведінка і взаємодія об’єктів реального світу (тобто в програмі створюються віртуальні аналоги реальних об’єктів).

Системне програмування з використанням WinAPI використовується для операційних систем Windows, а дана ОС є об’єктно-орієнтованою, і всі елементи управління (вікна) являються об’єктами. Кожен такий елемент має свої параметри, стан вхідні і вихідні повідомлення.

WinAPI – це складний функціональний інструмент, який обробляє деякі аспекти самостійно, наприклад, реєстрацію WNDCLASS. Тут кожен потік програми пов’язаний з локальним класом WNDCLASS, який формує всі можливі форми вікон, створені системою.

Об’єкти – це особливі програмні одиниці, що складаються з даних і алгоритмів для обробки саме цих даних. Дані, з яких складається об’єкт, називаються полями. Алгоритми, які входять в склад об’єкта, називаються методами.

Класи – це об’єктні типи даних. Всі об’єкти одного класу мають однаковий набір полів і однаковий набір методів. Важливою перевагою використання класів і об’єктів є те, що перевірка логічної сумісності даних і функцій для обробки даних є тривіальною задачею і може бути перекладена на компілятор – який тепер сам може визначити невірне використання даних.

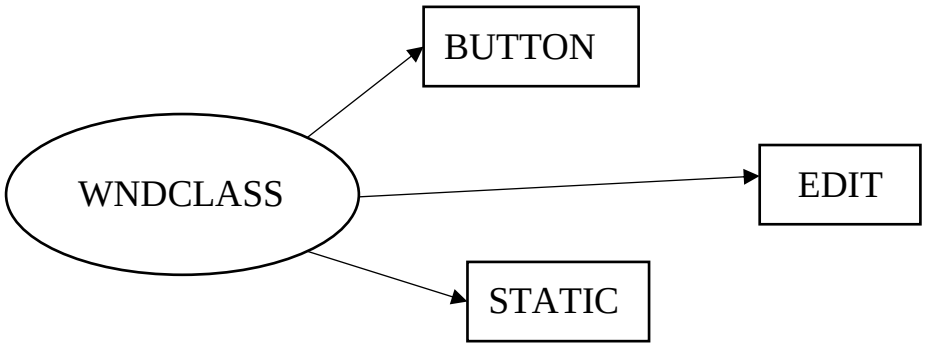


Рисунок 2.1 – Схема побудови графічного інтерфейсу

На рисунку 2.1 схематично вказаний головний клас і підкласи виконуваної задачі. WNDCLASS – це головний клас (головне вікно), а BUTTON, EDIT, STATIC дочірні вікна. Клас BUTTON використовується для створення кнопок натискання, EDIT – для полів редагування та введення інформації, STATIC – для виведення результатів розрахунку. Кнопки управляються за допомогою повідомлень.

2.2. Проектування алгоритму розв’язку задачі

Структура програмного додатку базується на діях користувача, тобто на виборі та послідовному введенні операцій. Дана структура вказана на рисунку 2.2.

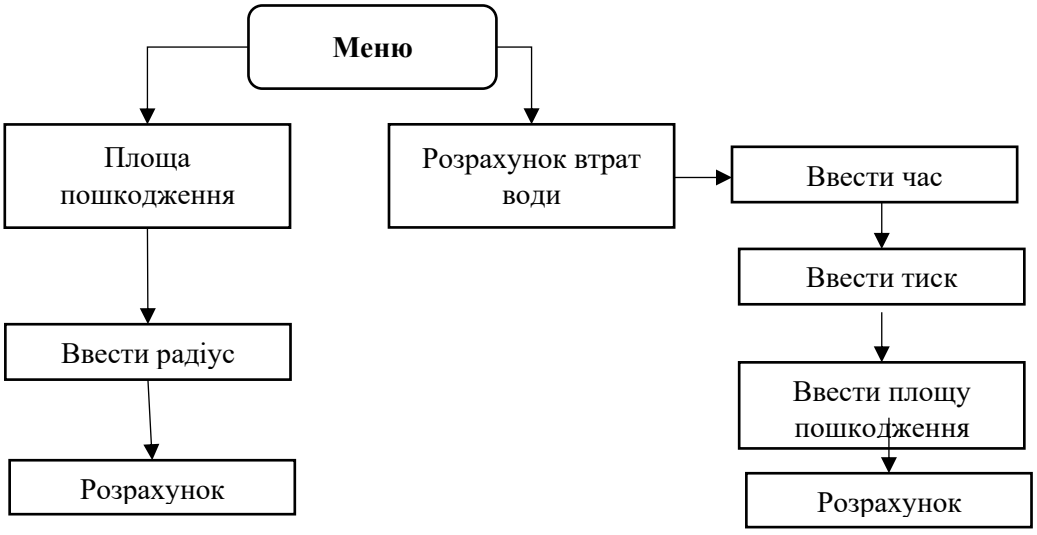


Рисунок 2.2. – загальна структура програми

Такий підхід спрощує створення програми і робить її зручною для кінцевого користувача. Меню користувача повинно бути інтуїтивно зрозумілим для всіх категорій працівників. Після запуску програми відкривається головне меню, де спочатку потрібно ввести радіус пошкодження і натиснути розрахунок. Для визначення розрахунку втрат води при пошкодженнях трубопроводу, потрібно ввести час тривалості аварії, робочий тиск та до цього розраховану площу пошкодження. Та натиснути «Розрахунок».

Загальну схему програми можна розділити на 7 частин. Перша – це підключення та створення глобальних змінних; друга - функція перетворення числа в символ; третя частина – створення головного класу вікна; четверта – створення класів вікон «EDIT», полів для введення інформації; п’ята частина - створення класів вікон «STATIC», кнопок з описом елементів; шоста частина – процедура обробки повідомлень; сьома – виконання розрахунків за формулами.

На рисунках 2.3, 2.4 вказано алгоритм перетворення числа в символ і навпаки. Такі перетворення реалізують стандартні бібліотечні функції, оголошені в <stdlib. h> (табл.2.3). Типи параметрів цих функцій наступні: const char *st; char **end; int base.

Ці функції відрізняються між собою тільки типом параметра num – цілого числа, яке потрібно перетворити. Всі три функції формують із числа num рядок символів, що відповідає запису цього числа в системі числення з основою base, і повертають вказівник на перший символ створеного рядка.

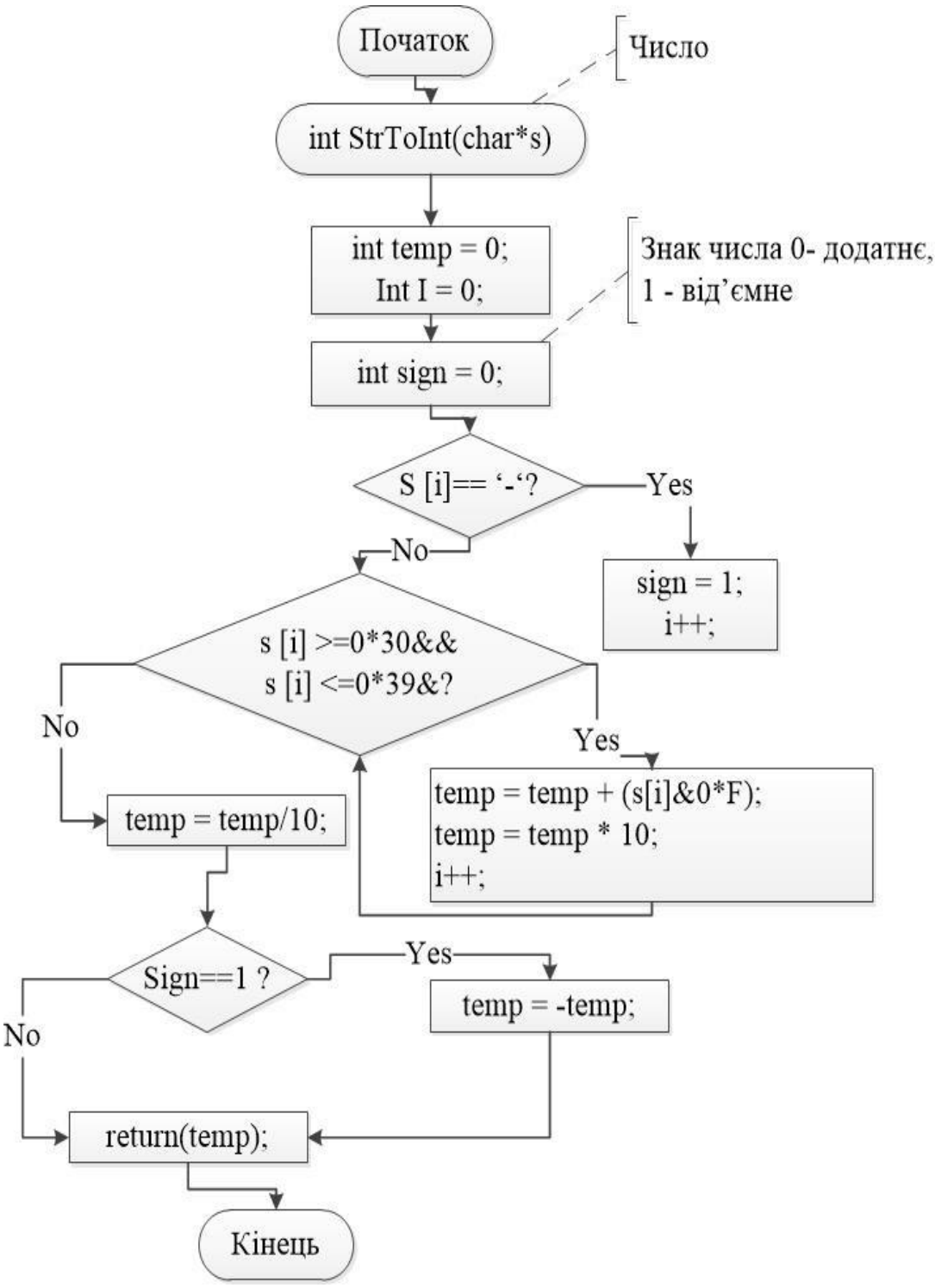


Рисунок - 2.3 Блок-схема алгоритму функції перетворення числа в символ

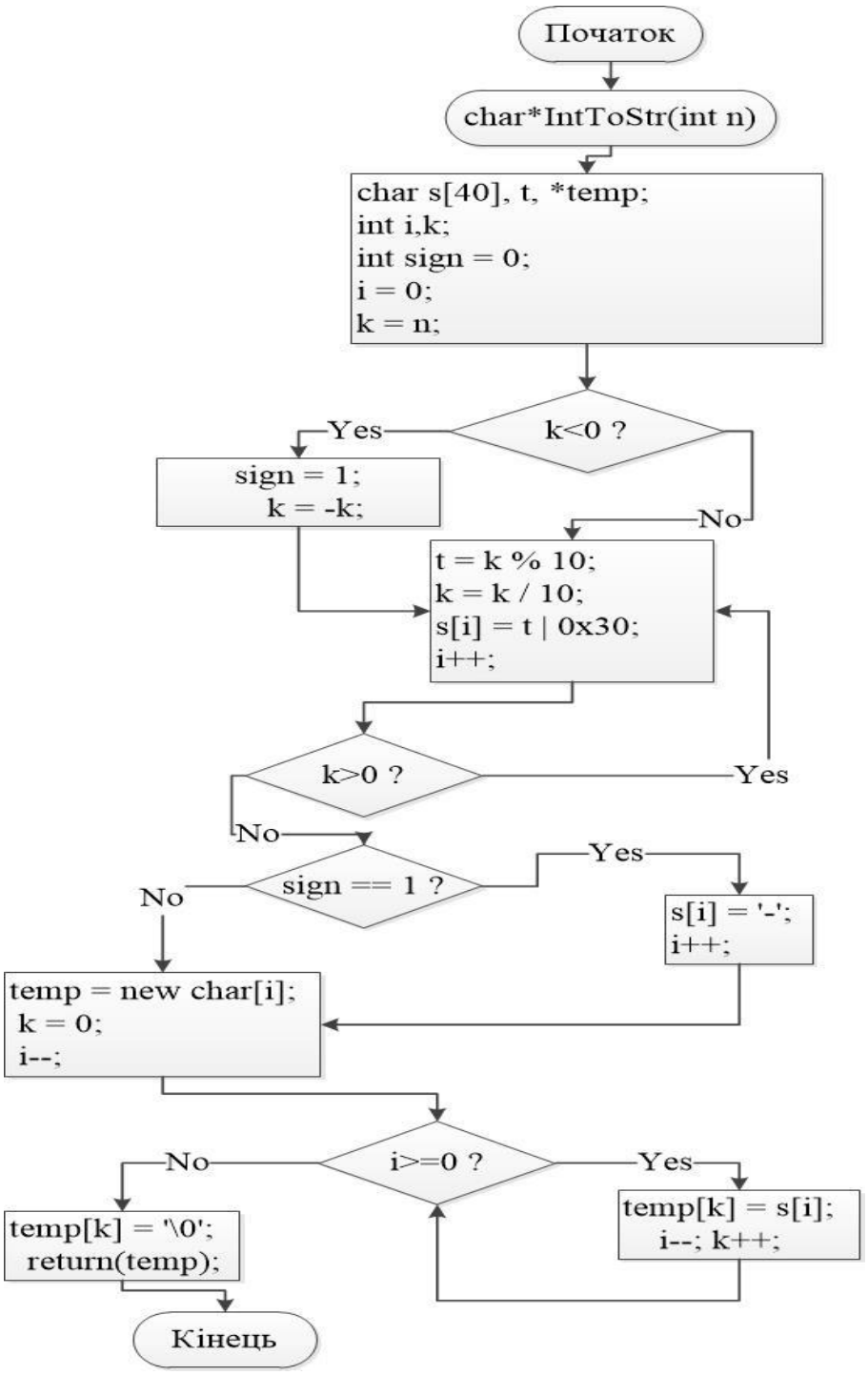
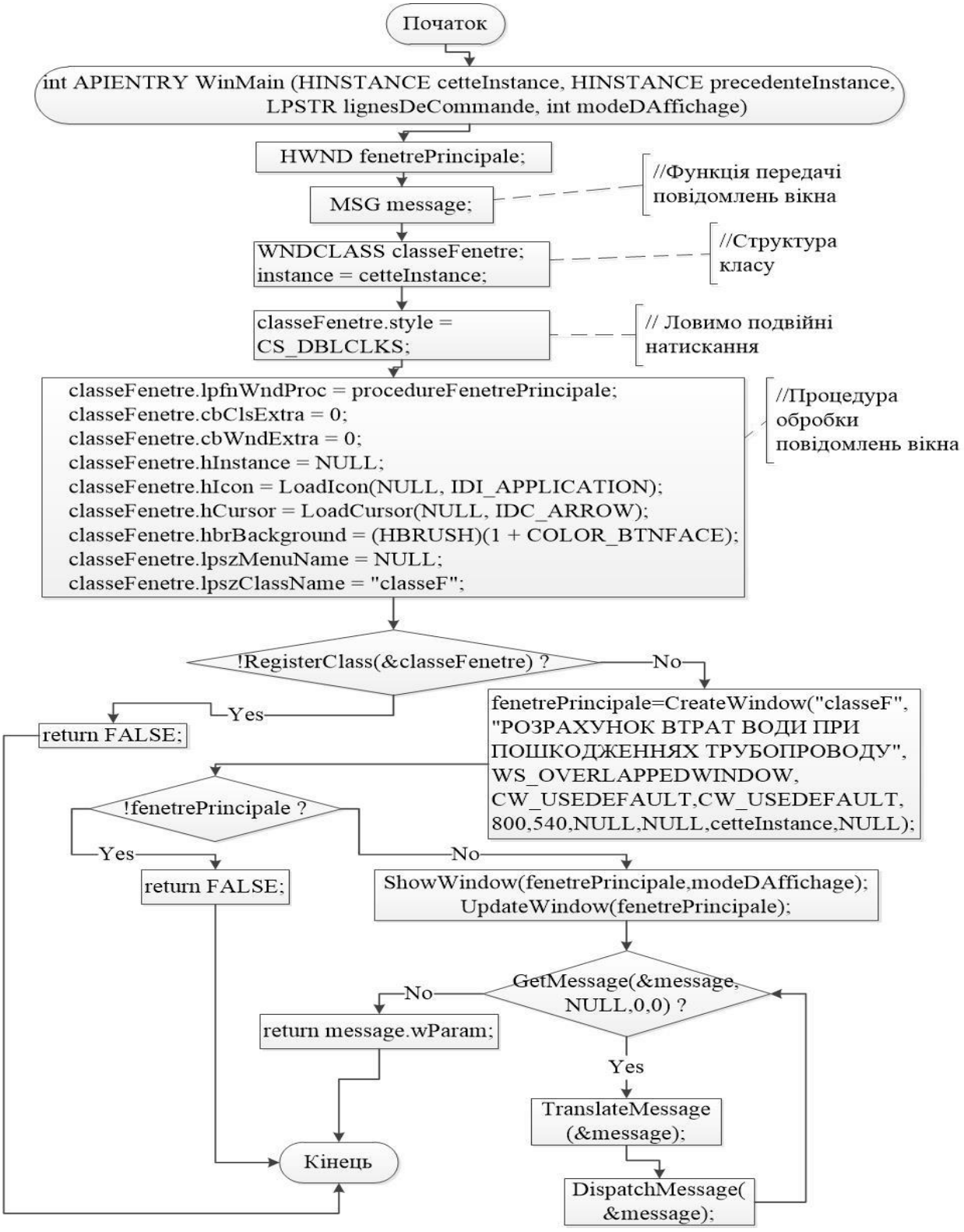


Рисунок - 2.4 Блок-схема алгоритму функції перетворення символу в число

Алгоритм створення головного класу вікна (рис.2.5) полягає в: створенні та реєстрації класу вікна, створення вікна (вказуємо розмір та координати, вказуємо ім’я класу вікна, стиль, заголовок), показ вікна, створюємо функцію вибірки повідомлень та зберігаємо їх в змінну message,

створюємо функцію передачі повідомлень у вікно та створюємо процедуру



обробки повідомлень і зчитуємо подвійні натискання.

Рисунок - 2.5 Блок-схема алгоритму створення загального класу вікна та процедури обробки повідомлень

На рисунку 2.6 вказаний алгоритм створення класів вікон «EDIT»,

для введення: періоду тривалості аварії, робочого тиску, радіусу пошкодження, площі пошкодження виведення результату розрахунку площі пошкодження та загального розрахунку.

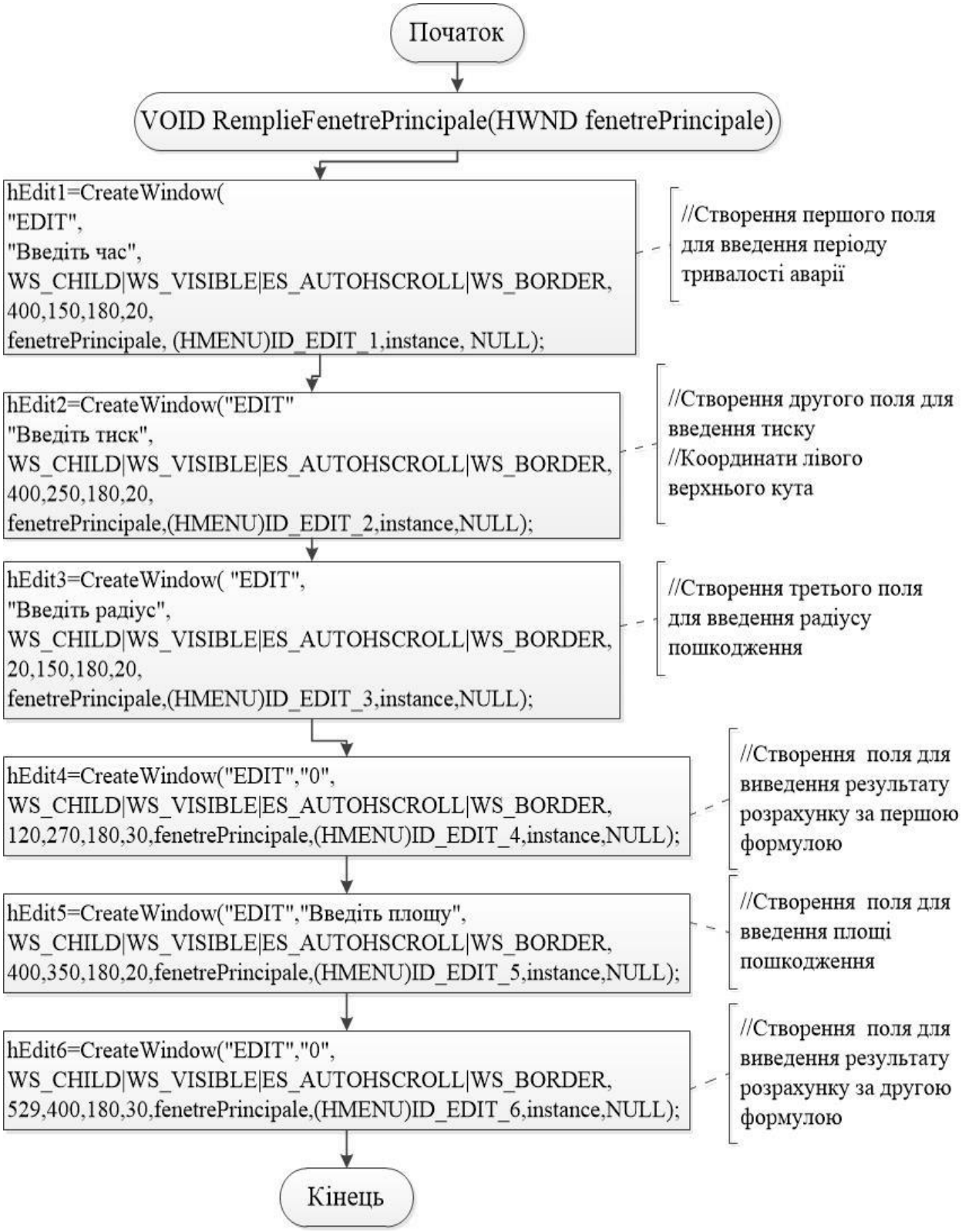


Рисунок - 2.6 Блок-схема алгоритму створення кнопок класу «Edit»

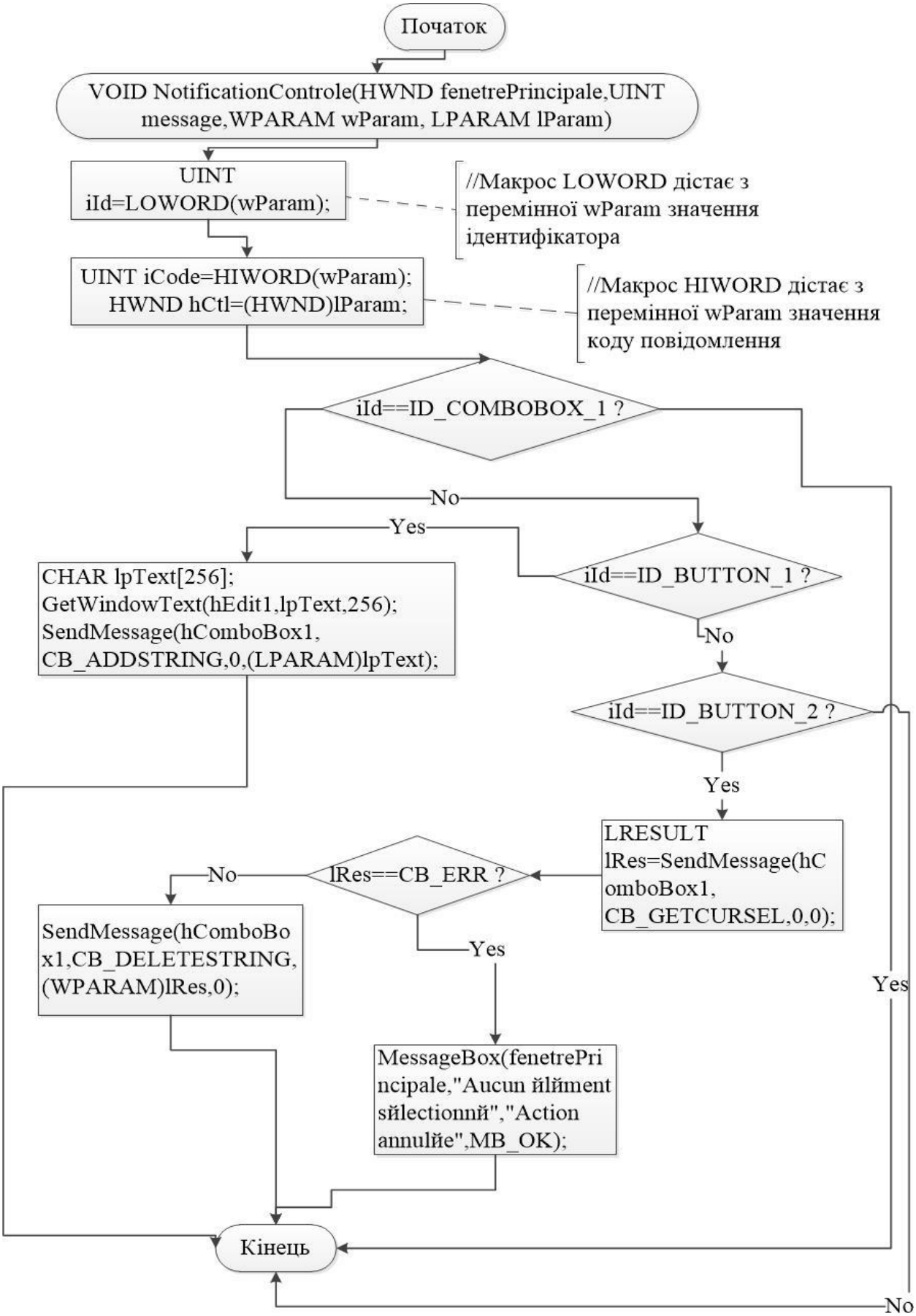


Рисунок - 2.7 Блок-схема алгоритму створення процедури обробки повідомлень

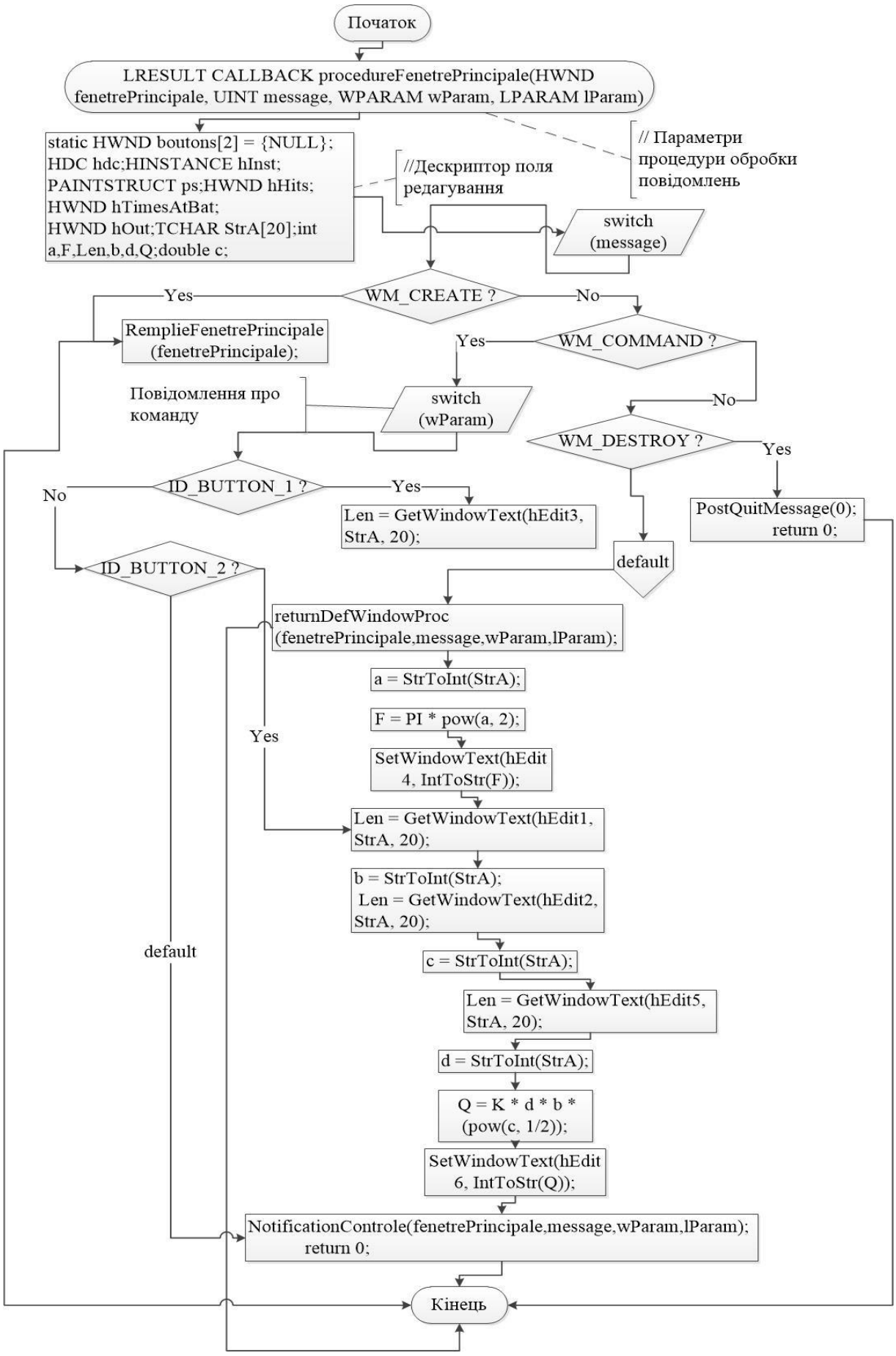


Рисунок - 2.8 Блок-схема алгоритму задання параметрів процедури

На рисунках 2.7 – 2.8 вказані блок-схеми алгоритму задання параметрів процедури обробки повідомлень. Тобто макрос LOWORD дістає з перемінної wParam значення ідентифікатора та дістає з перемінної wParam значення коду повідомлення. Потім задаємо команду (case WM_ COMMAND), що при натисканні на кнопку 1 (BUTTON_1) необхідно зчитати число з першого поля та порахувати за формулою площу пошкодження. Далі – вивести результат в поле 4. При натисканні на кнопку 2 (BUTTON_2), зчитуємо число з першого та другого поля.

Функція GetWindowText копіює текст заданого об'єкту у вказаний буфер перший параметр-це адреса об'єкту, другий параметр - це адреса буферу, останній параметр - це максимальна кількість копійованих символів.

Далі зчитуємо число з третього поля та знаходимо за формулою (Q = K * d * b * (pow(c, 1/2))) обсяг втрат води при пошкодженнях трубопроводу і виводимо результат в поле 6.

Команда case WM_DESTROY: PostQuitMessage(0); return 0; закриває вікно. А команда default: return DefWindowProc(fenetrePrincipale,message,wParam,lParam);} обробляє повідомлення за змовчуванням.

Отже, враховуючи особливості структури програми та поставлені завдання запропонований алгоритм є оптимальним в даному випадку. Він дозволяє зробити програму зрозумілою та структурованою, також при необхідності програма може бути доповнена новими модулями.

РОЗДІЛ 3. ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ

3.1. Опис головних структур і змінних програми

Меню розробленого програмного забезпечення представлене у вигляді графічного інтерфейсу з інтуїтивно зрозумілим інтерфейсом. Використовуючи підказки інтерфейсу користувач може перейти до необхідного пункту меню. Інтерфейс продумано так, що навіть знаючи площу пошкодження трубопроводу, можна одразу порахувати обсяг втрат води. А можна й рахувати спочатку площу пошкодження, а потім розрахунок втрат. Структура меню вказана на рисунку 3.1.

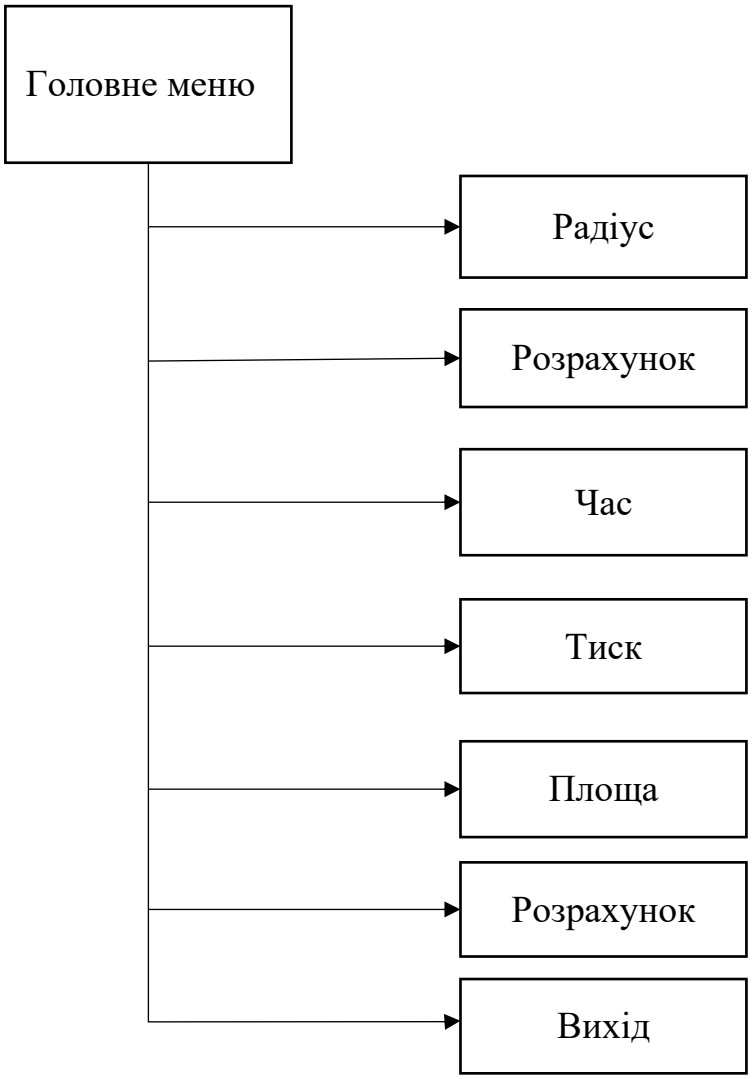


Рисунок 3.1 – Структура меню програми

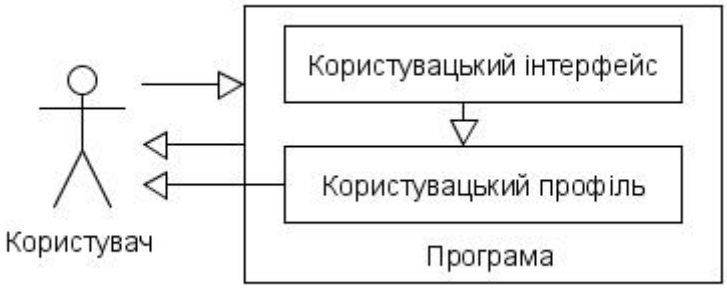


Рисунок 3.2 – Схема взаємодії програми з користувачем

На рисунку 3.2 схематично вказано зовнішній вигляд користувацького інтерфейсу. який залежить від представленого профілю користувача. Профіль користувача включає інформацію необхідну для користування програмою.

Як бачимо, графічний інтерфейс та меню програми є зрозумілим та простим у використанні. Є все необхідне для забезпечення зручного робочого процесу та швидкого розрахунку.

3.2. Описання реалізованих функцій

У таблиці 3.1 вказано описання всіх функцій, використаних в написанні програми та їх призначення.

Таблиця 3.1

Опис функцій головного меню програми

№П/П	Прототип функції	Призначення функції
1.	memset(&wcl, 0, sizeof(WNDCLASSA));	функція memset заповнює всі байти по вказаній адресі, в даному випадку заповнюємо все нулями. Таким чином присвоїли всім байтам структури wcl значення нуль.
2.	wcl.IpfnWndProc = DefWindowProcA	функція обробки повідомлень за змовчуванням.
3.	CreateWindow	функція CreateWindow створює вікно та повертає дескриптор вікна.
4.	RegisterClass	Реєструє клас вікна.

Тип WNDCLASSA являється структурою класу і для реєстрації класу необхідно його проініціалізувати. Для цього необхідно попередньо заповнити змінну нулями за допомогою команди: memset(&wcl, 0, sizeof(WNDCLASSA)); функція memset заповнює всі байти по вказаній адресі, в даному випадку заповнюємо все нулями. Таким чином присвоїли всім байтам структури wcl значення нуль.

Функція GetMessage дістає повідомлення з черги повідомлень і зберігає його в змінну типу msg. Якщо повідомлень в черзі немає, то функція буде очікувати. І цикл буде виконуватись лише тоді, коли буде отримано повідомлення з черги, в якості результату функція верне True. Команда DispatchMessage(&msg); передає отримане повідомлення нашому вікну.

В таблиці 3.2 вказано опис функцій, що забезпечують реалізацію програми.

Таблиця 3.2

Опис функцій, що забезпечують реалізацію програми

№П/П	**Прототип функції**	**Призначення функції**
1.	GetMessage	Функція GetMessage дістає повідомлення з черги повідомлень і зберігає його в змінну типу msg. Якщо повідомлень в черзі немає, то функція буде очікувати. І цикл буде виконуватись лише тоді, коли буде отримано повідомлення з черги, в якості результату функція верне True. Команда DispatchMessage(&msg); передає отримане повідомлення нашому вікну.
2.	GetClassInfoExA (winuser.h)	Витягує інформацію про клас вікна, включаючи дескриптор невеликого

		значка, пов'язаного з класом вікна.

Таблиця 3.3

Функції перетворення символьних рядків у числа

№П/П	**Прототип функції**	**Призначення функції**
1.	int atoi (st);	Виділяє у рядку st перше ціле десяткове число і перетворює його у дане з типом int.
2.	long atol (st) ;	Аналог atoi(), але перетворює рядок у число з типом long.
3.	double atof (st);	Аналог atoi(), але перетворює рядок st у дане з типом double.
4.	long strtol (st, end, base);	Розширений варіант atol().
5.	unsigned long strtoul(st, end, base);	Аналог strtol(), але повертає значення, що має тип unsigned long.
6.	fcouble strtod (st, end);	Розширений варіант atof(). Перетворює початкову частину st у дане з типом double. Додатково повертає через end адресу першого символу, записаного за числом.
7.	double strtof (st, end);*	Аналог strtod(), але повертає значення з типом float.
8.	double strtold (st, end);*	Аналог strtod(), але повертає значення, що має тип long double.

Всі вищевказані функції призначені для реалізації робочого процесу.

Вони забезпечують: отримання бажаної розмірності поля програми від користувача, заповнення полів необхідною інформацією та точним розрахунком за формулами.

Деякі функції логічно пов’язані між собою. Для зручності структурування їх було виокремлено в формули.

3.3. Опис інтерфейсу

Для того, щоб здійснити розрахунок втрат води при пошкодженнях трубопроводу в новій програмі потрібно завантажити файл RVT.exe та відкрити його подвійним натисканням миші.

Розроблюваний програмний продукт реалізовано у вигляді графічного інтерфейсу типу класичного калькулятора. Після запуску програми відкриється головне меню, з кнопками та полями редагування куди необхідно ввести дані про пошкодження трубопроводу (Рис.3.3).

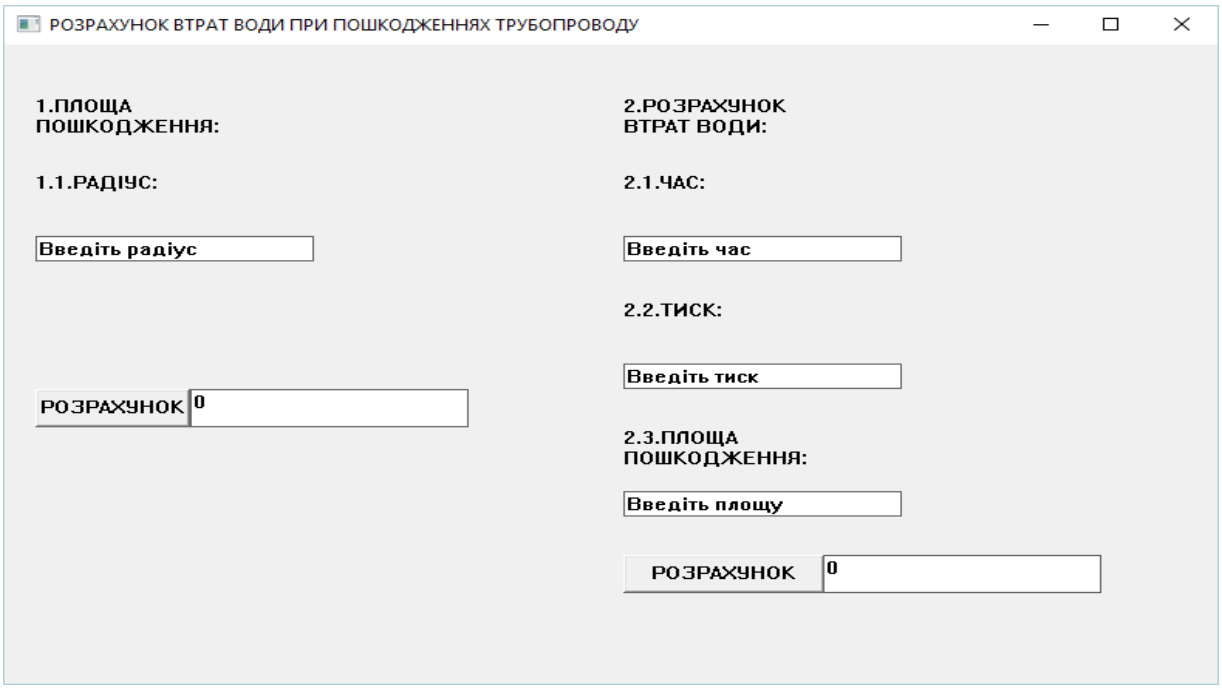


Рисунок 3.3 – Головне меню програми

Покрокова інструкція розрахунку:

1. Запускаємо файл RVT.exe подвійним натисканням миші.
2. В поле «Введіть радіус» вводимо радіус пошкодження трубопроводу.

3. Натискаємо «Розрахунок».

4. Вводимо час тривалості аварії.

5. Вводимо робочий тиск.

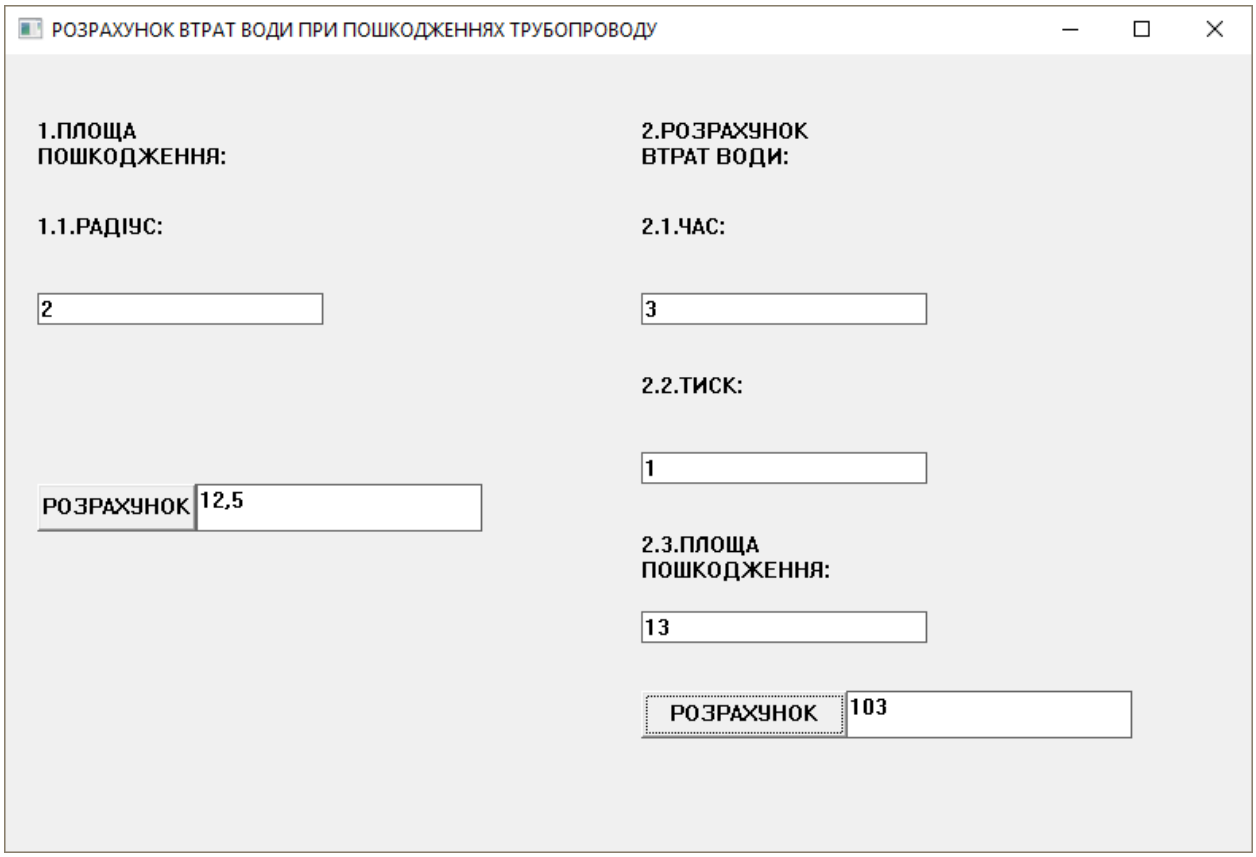
6. Вводимо площу пошкодження пораховану в першій частині.

7. Натискаємо «Розрахунок».

8. Результат виводиться в вікно, біля кнопки «Розрахунок».

Отже, розроблене програмне забезпечення має зручний та зрозумілий графічний інтерфейс користувача. Меню програми є досить простим, проте у ньому є все необхідне для виконуваної задачі.

3.4. Результати роботи програмного продукту



Результати роботи програми перевірено на контрольних прикладах. Розрахунки виконано як і в розробленій програмі, так і в Excel. На рисунку 3.4 вказано результат перевірки розрахунків самої програми.

Рисунок 3.3 – Результат перевірки правильності розрахунків

Вводимо радіус пошкодження 2, натискаємо на кнопку «Розрахунок», і в результаті програма рахує за формулою знаходження площі пошкодження: F = PI * pow(a, 2). Далі вводимо час тривалості аварії – 3 години, робочий тиск – 1, та переносимо площу пошкодження, пораховану за першою формулою – 12. Натискаємо на кнопку «Розрахунок». Програма рахує за формулою Q = K * d * b * (pow(c, 1/2)). Результатом розрахунку є 131. Тобто, під час аварії на основних водопроводах тривалістю 3 години втрачено 131 кубів води.

Перевіримо дані розрахунки в Excel. Результат вказаний на рисунку 3.4.

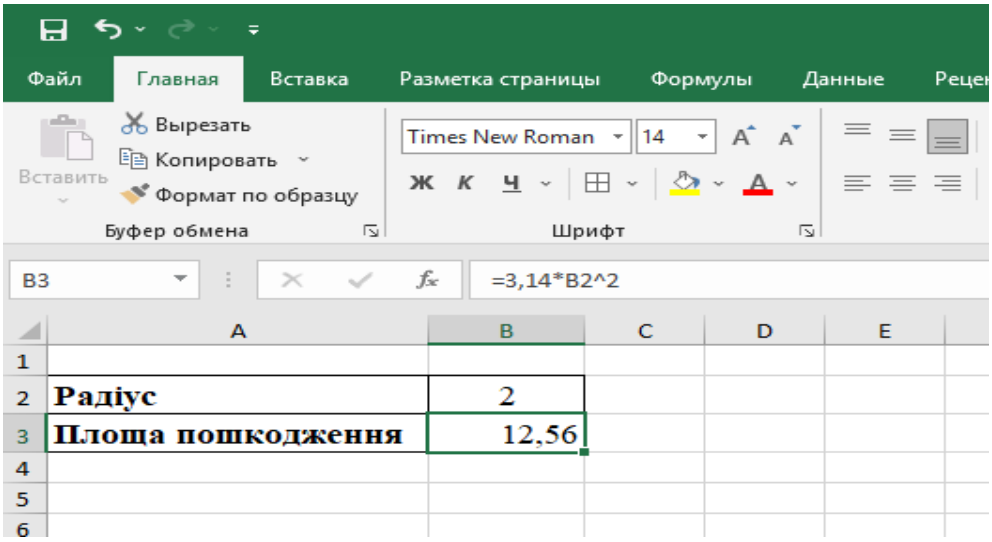


Рисунок 3.4 – Перевірка розрахунку площі пошкодження

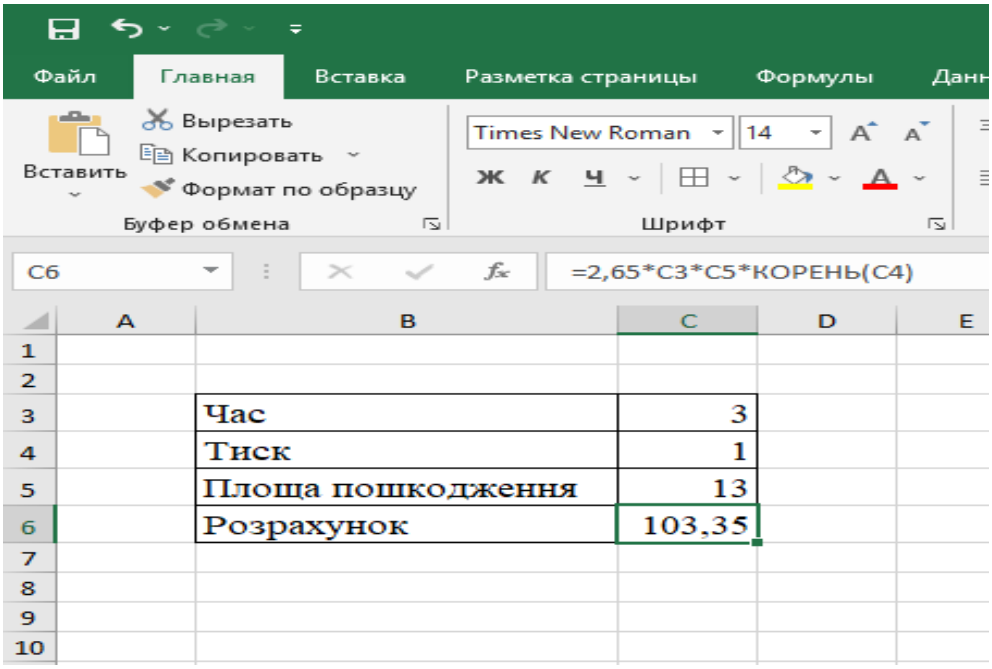


Рисунок 3.5 – Перевірка розрахунку загального розрахунку

Отже, як показали результати тестування, програма працює коректно, графічний інтерфейс є зручним та інтуїтивно зрозумілим.

Отже, всі вимоги технічного завдання виконані.

ВИСНОВКИ

В результаті виконання курсової роботи вирішено задачу автоматизації розрахунку втрат води при пошкодженнях трубопроводу в департаментах Київводоканал. Для цього створено та реалізовано програму з користувацьким інтерфейсом.

У першому розділі виконано огляд наукової літератури, розглянуто програми аналоги розрахунку втрат води, обґрунтовано вибір мови програмування та середовище розроблення, також розглянуто компілятори аналоги, де вказані всі переваги та недоліки. Створено постановку задачі розроблення програми.

У другому розділі розглянуто математичну основу алгоритмів та встановлено етапи розв’язку задачі у вигляді блок-схем алгоритмів.

У третьому розділі описано основну програму, створено інструкцію користувача, та протестовано результати розрахунків та правильності роботи програми.

Програму створено в середовищі розроблення Code::Blocks 20.03 з використанням мови програмування C++. Результатом роботи програми є автоматичний розрахунок за формулами в зручному користувацькому інтерфейсі. Програму протестовано на правильність розрахунків.

Всі пункти технічного завдання виконані в повному обсязі. В подальшому програму можна вдосконалити з подальшим додаванням інших формул та зміні користувацького інтерфейсу за вимогою інженерів водоканалу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Щупак Ю.А. Win32 API Эффективная разработка приложений. Питер. 2007.

2. Щупак Ю.А. Win32 API Разработка приложений для Windows. Питер, 2008.

3. Андрощук О. Комп'ютерні ігри як засіб розвитку творчих здібностей старших дошкільників. *Вісник Інституту розвитку дитини. Сер.: Філософія, педагогіка, психологія.* 2014. Вип. 34. С. 137-142. URL: http://nbuv.gov.ua/UJRN/Vird_2014_34_23 (дата звернення: 02.11.2020)

4. Вінник В.Ю. Алгоритмічні мови та основи програмування: мова С.: навч. посіб. Житомир: ЖДТУ, 2007. 328 с.

5. Брайан У. Керниган, Деннис М. Ритчи. Язык программирования С, второе издание: уч. пособ. Москва: Диалектика-Вильямс, 2016. 304 с.

6. Шпак З.Я. Програмування мовою С: навч. посіб. Львів: Оріяна-Нова, 2006. 432 с.

7. Xcode IDE. URL: https://developer.apple.com/xcode/features/ (дата звернення: 06.11.2020)

8. Калужнин Л. А., Сущанский В. И. Преобразования и перестановки: пер. с укр. 2-е изд., Москва: Наука. Главная редакция физико-математической литературы, 1985. 160 с.

9. Тичинська Л. М., Черепащук А. А. Теорія ймовірностей. ч. 1. Історичні екскурси та основні теоретичні відомості: навч. посіб. Вінниця: ВНТУ, 2010. 112 с.

10. Гарднер М. Математические досуги: учеб. пособ. Москва: Мир, 1972. [Глава 33 «Игра в 15 и другие головоломки»]. С.402

11. Андреев Н. Н., Коновалов С. П., Панюнин Н. М. Математическая составляющая, 2-е изд., Москва: Фонд «Математические этюды», 2019. 367с.

12. Толстиков А.В. Алгебра и геометрия: учеб. пособ., Череповец: ГОУ ВПО Череповец. гос. ун-т, 2004. 321с.

ДОДАТКИ

ДОДАТОК А

Лістинг програми

```
#include <windows.h> // Підключаємо модуль Windows
#include <stdio.h>  // Стандартна бібліотека введення-виведення
#include <math.h>  //Підключаємо модуль математичних функцій



HINSTANCE instance;

#define ID_COMBOBOX_1       110 //Ідентифікатори для кнопок
#define ID_BUTTON_1         111
#define ID_BUTTON_2         112
#define ID_EDIT_1           113
#define ID_EDIT_2           114
#define ID_EDIT_3           115
#define ID_STATIC_1         116
#define ID_EDIT_4           117
#define ID_EDIT_5           118
#define ID_EDIT_6           119
#define PI 3.142
#define K 2.65

HWND hComboBox1; //Створення глобальних змінних
HWND hEdit1;
HWND hEdit2;
HWND hEdit3;
HWND hEdit4;
HWND hEdit5;
HWND hEdit6;
```

```
int StrToInt(char *s)
{
  int temp = 0; // число
  int i = 0;
  int sign = 0; // знак числа 0- додатнє, 1 - від'ємне
  if (s[i] == '-')
  {
    sign = 1;
    i++;
  }
  while (s[i] >= 0x30 && s[i] <= 0x39)
  {
    temp = temp + (s[i] & 0x0F);
    temp = temp * 10;
    i++;
  }
  temp = temp / 10;
  if (sign == 1)
    temp = -temp;
  return(temp);
}
// Функція перетворення числа в символ
char* IntToStr(int n)
{
  char s[40], t, *temp;
  int i, k;
  int sign = 0;
  i = 0;
  k = n;
```

```
 if (k<0)
 {
  sign = 1;
  k = -k;
 }
 do {
  t = k % 10;
  k = k / 10;
  s[i] = t | 0x30;
  i++;
 } while (k>0);
 if (sign == 1)
 {
  s[i] = '-';
  i++;
 }
 temp = new char[i];
 k = 0;
 i--;
 while (i >= 0) {
  temp[k] = s[i];
  i--; k++;
 }
 temp[k] = '\0';
 return(temp);
}

VOID RemplieFenetrePrincipale(HWND fenetrePrincipale)
{
```

```
hEdit1=CreateWindow(     //Створення першого поля для введення періоду тривалості аварії
"EDIT",
"Введіть час",
WS_CHILD|WS_VISIBLE|ES_AUTOHSCROLL|WS_BORDER,
400,150,                  //Координати лівого верхнього кута
180,20,                   //Розмір кнопки
fenetrePrincipale,
(HMENU)ID_EDIT_1,
instance,
NULL);

hEdit2=CreateWindow( //Створення другого поля для введення тиску
"EDIT",
"Введіть тиск",
WS_CHILD|WS_VISIBLE|ES_AUTOHSCROLL|WS_BORDER,
400,250,                  //Координати лівого верхнього кута
180,20,                   //Розмір кнопки
fenetrePrincipale,
(HMENU)ID_EDIT_2,
instance,
NULL);

hEdit3=CreateWindow(  //Створення третього поля для введення радіусу пошкодження
"EDIT",
"Введіть радіус",
WS_CHILD|WS_VISIBLE|ES_AUTOHSCROLL|WS_BORDER,
```

```
//Флаг WS_BORDER додає границю до нашого елементу
            20,150,              //Координати лівого верхнього кута
            180,20,               //Розмір кнопки
            fenetrePrincipale,
            (HMENU)ID_EDIT_3,
            instance,
            NULL);

            hEdit4=CreateWindow(  //Створення  поля для виведення результату розрахунку за першою формулою
            "EDIT",
            "0",
            WS_CHILD|WS_VISIBLE|ES_AUTOHSCROLL|WS_BORDER,
            120,270,            //Координати лівого верхнього кута
            180,30,             //Розмір кнопки
            fenetrePrincipale,
            (HMENU)ID_EDIT_4,
            instance,
            NULL);

            hEdit5=CreateWindow(  //Створення  поля для введення площі пошкодження
            "EDIT",
            "Введіть площу",
            WS_CHILD|WS_VISIBLE|ES_AUTOHSCROLL|WS_BORDER,
            400,350,            //Координати лівого верхнього кута
            180,20,            //Розмір кнопки
            fenetrePrincipale,
            (HMENU)ID_EDIT_5,
            instance,
```

```
NULL);

hEdit6=CreateWindow(  //Створення  поля для виведення
результату розрахунку за другою формулою
"EDIT",
"0",
WS_CHILD|WS_VISIBLE|ES_AUTOHSCROLL|WS_BORDER,
529,400,              //Координати лівого верхнього кута
180,30,              //Розмір кнопки
fenetrePrincipale,
(HMENU)ID_EDIT_6,
instance,
NULL);


CreateWindow("STATIC","2.1.ЧАС:",WS_VISIBLE|WS_CHILD,400,100,150,3
8,
              fenetrePrincipale,
(HMENU)ID_STATIC_1,
instance,
NULL);


CreateWindow("STATIC","2.2.ТИСК:",WS_VISIBLE|WS_CHILD,400,200,150,
38,
              fenetrePrincipale,
(HMENU)ID_STATIC_1,
instance,
NULL);
```

```
        CreateWindow("STATIC","2.3.ПЛОЩА
ПОШКОДЖЕННЯ:",WS_VISIBLE|WS_CHILD,400,300,150,38,
                fenetrePrincipale,
        (HMENU)ID_STATIC_1,
        instance,
        NULL);

        CreateWindow("STATIC","1.ПЛОЩА
ПОШКОДЖЕННЯ:",WS_VISIBLE|WS_CHILD,20,40,150,38,
                fenetrePrincipale,
        (HMENU)ID_STATIC_1,
        instance,
        NULL);

        CreateWindow("STATIC","2.РОЗРАХУНОК ВТРАТ
ВОДИ:",WS_VISIBLE|WS_CHILD,400,40,150,38,
                fenetrePrincipale,
        (HMENU)ID_STATIC_1,
        instance,
        NULL);


CreateWindow("STATIC","1.1.РАДІУС:",WS_VISIBLE|WS_CHILD,20,100,150
,38,
                fenetrePrincipale,
        (HMENU)ID_STATIC_1,
        instance,
        NULL);

      CreateWindow(  //Створення кнопки розрахунку площі пошкодження
```

```
(F)
            "BUTTON",
            "РОЗРАХУНОК",
            WS_CHILD|WS_VISIBLE|BS_PUSHBUTTON, //Флаги: дочірнє вікно, показ кнопки,
            20,270,  //Координати верхнього лівого вікна кнопки
            100,30,  //Розмір кнопки
            fenetrePrincipale,
            (HMENU)ID_BUTTON_1,
            instance,
            NULL);

        CreateWindow( //Створення кнопки розрахунку втрат води при пошкодженні трубопроводу(Q)
            "BUTTON",
            "РОЗРАХУНОК",
            WS_CHILD|WS_VISIBLE|BS_PUSHBUTTON,
            400,400,
            130,30,
            fenetrePrincipale,
            (HMENU)ID_BUTTON_2,
            instance,
            NULL);
    }

    VOID NotificationControle(HWND fenetrePrincipale,UINT message,WPARAM wParam, LPARAM lParam)
    {
        UINT iId=LOWORD(wParam); //Макрос LOWORD дістає з перемінної wParam значення ідентифікатора
```

```
        UINT iCode=HIWORD(wParam); //Макрос HIWORD дістає з
перемінної wParam значення коду повідомлення
        HWND hCtl=(HWND)lParam;

       if(iId==ID_COMBOBOX_1)
       {

       }
       else if(iId==ID_BUTTON_1)
       {
          CHAR lpText[256];
          GetWindowText(hEdit1,lpText,256);

          SendMessage(hComboBox1,CB_ADDSTRING,0,(LPARAM)lpText);
       }
       else if(iId==ID_BUTTON_2)
       {
         LRESULT lRes=SendMessage(hComboBox1,CB_GETCURSEL,0,0);
         if(lRes==CB_ERR)
         {
            MessageBox(fenetrePrincipale,"Aucun йlйment sйlectionnй","Action
annulйe",MB_OK);
            return ;
         }


SendMessage(hComboBox1,CB_DELETESTRING,(WPARAM)lRes,0);
       }
     }
     // Параметри процедури обробки повідомлень
```

```
LRESULT CALLBACK procedureFenetrePrincipale(HWND
fenetrePrincipale, UINT message, WPARAM wParam, LPARAM lParam)

{
   static HWND boutons[2] = {NULL};
   HDC hdc;
   HINSTANCE hInst;
   PAINTSTRUCT ps;
   HWND hHits;
   HWND hTimesAtBat; //Дескриптор поля редагування
   HWND hOut;
   TCHAR StrA[20];
   int a,F,Len,b,d,Q;
   double c;




   switch (message)
   {
      case WM_CREATE:
         RemplieFenetrePrincipale(fenetrePrincipale);
         return 0;


               case WM_COMMAND: // Повідомлення про команду


       switch(wParam)
      {
```

```
                case ID_BUTTON_1: //При натисканні кнопки:
        Len = GetWindowText(hEdit3, StrA, 20);
        a = StrToInt(StrA); // Зчитуємо число з першого поля

        F = PI * pow(a, 2);  // Рахуємо за формулою площу пошкодження

            SetWindowText(hEdit4, IntToStr(F)); //Виводимо результат в
поле 4

            case ID_BUTTON_2: // При натисканні кнопки розрахунок втрат:
        Len = GetWindowText(hEdit1, StrA, 20);
        b = StrToInt(StrA); // Зчитуємо число з першого поля

        Len = GetWindowText(hEdit2, StrA, 20);
        c = StrToInt(StrA); // Зчитуємо число з другого поля

        Len = GetWindowText(hEdit5, StrA, 20);/*Функція GetWindowText
копіює текст заданого об'єкту у вказаний буфер
        перший параметр-це адреса об'єкту, другий параметр - це адреса
буферу,
            останнфй параметр - це максимальна кількість копійованих
символів*/
        d = StrToInt(StrA); // Зчитуємо число з третього поля

        Q = K * d * b * (pow(c, 1/2));  // Знаходимо за формулою обсяг втрат
води при пошкодженнях трубопроводу

            SetWindowText(hEdit6, IntToStr(Q)); //Виводимо результат в
поле 6
```

```
            break;

        }
            NotificationControle(fenetrePrincipale,message,wParam,lParam);
            return 0;


                case WM_DESTROY: // Закриття вікна
            PostQuitMessage(0);
            return 0;
        default:  //  Оброблення повідомлень за змовчуванням
            return DefWindowProc(fenetrePrincipale,message,wParam,lParam);
    }
}


int APIENTRY WinMain (HINSTANCE cetteInstance, HINSTANCE precedenteInstance,
                LPSTR lignesDeCommande, int modeDAffichage)
{
    HWND fenetrePrincipale;
    MSG message; //Функція передачі повідомлень вікна
    WNDCLASS classeFenetre;  //Структура класу

    instance = cetteInstance;

    classeFenetre.style = CS_DBLCLKS;// Ловимо подвійні натискання
    classeFenetre.lpfnWndProc = procedureFenetrePrincipale; //Процедура обробки повідомлень вікна
```

```
classeFenetre.cbClsExtra = 0;
classeFenetre.cbWndExtra = 0;
classeFenetre.hInstance = NULL;
classeFenetre.hIcon = LoadIcon(NULL, IDI_APPLICATION);
classeFenetre.hCursor = LoadCursor(NULL, IDC_ARROW);
classeFenetre.hbrBackground = (HBRUSH)(1 + COLOR_BTNFACE);
classeFenetre.lpszMenuName = NULL;
classeFenetre.lpszClassName = "classeF";


if(!RegisterClass(&classeFenetre)) //Створюємо та реєструємо клас
вікна
   return FALSE;

fenetrePrincipale=CreateWindow(   //Створення головного вікна
   "classeF",   //Ім'я класу вікна
   "РОЗРАХУНОК ВТРАТ ВОДИ ПРИ ПОШКОДЖЕННЯХ
ТРУБОПРОВОДУ",  //Заголовок вікна
   WS_OVERLAPPEDWINDOW,      // Стиль головного вікна
   CW_USEDEFAULT,CW_USEDEFAULT,
   800,540,  //Розмір вікна
   NULL,
   NULL,
   cetteInstance,
   NULL);

if (!fenetrePrincipale)
   return FALSE;

ShowWindow(fenetrePrincipale,modeDAffichage); //Показуємо вікно
```

```
        UpdateWindow(fenetrePrincipale);


        while(GetMessage(&message,NULL,0,0)) //фуккція вибірки
повідомлень та їх збереження в змінну message
        {
           TranslateMessage(&message);
           DispatchMessage(&message); //Функція передачі повідомлень у
вікно
        }

        return message.wParam;
    }
```

ДОДАТОК Б

Таблиця 1.1.

Порівняльна характеристика програм аналогів

№ П/П	Назва програми	Функції	Порівняння з програмою курсової
1.	Втрати тиску – розрахункова лінійка	Визначення втрати тиску в залежності від внутрішнього діаметра прямої труби, обсягів подачі, значення коефіцієнта «k»	Виконує розрахунок втрат води при пошкодженнях трубопроводу. До її створення розрахунок відбувався вручну на калькуляторі.
2.	«Pump & Boosters Selection Program»	Призначена для підбору насосів і гідровузлів (насосних станцій) фірми DP Pumps	
3.	Smedegaard Pump Selection system WPSEL	Призначена для оптимального підбору циркуляційних насосів. Циркуляційні насоси традиційно використовуються при створенні систем кондиціонування, систем опалення, систем холодного і гарячого водопостачання, для відводу стічних вод і в інших різних системах.	
4.	«Насос (Pump)» v. 1.5.0.0 Alpha	Аналіз базових (паспортних) параметрів насосних агрегатів (витрата води - Q, напір - H, частота обертання - n і діаметр - D робочого колеса), його характеристик (QH, QN, Q-к.к. д) і розрахунок нових параметрів при зміні будь-якого з базових.	