Міністерство освіти і науки України
Український державний університет науки і технологій

Факультет «Комп'ютерні технології і системи»

Кафедра «Комп'ютерні інформаційні технології»

Пояснювальна записка

до кваліфікаційної роботи бакалавра

на тему: «Гребінкова фільтрація часових рядів»
за освітньою програмою: «Інженерія програмного забезпечення»
зі спеціальності: «121 Інженерія програмного забезпечення»
Виконав: студент групи «ПЗ1812»

_____ /Володимир ПОГРЕБНЯК/
(підпис студента) (Ім'я ПРІЗВИЩЕ)

Керівник: _____ / Віктор ШИНКАРЕНКО/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Нормоконтролер: _____ /доц. Олена КУРОП'ЯТНИК/
(підпис) (посада, Ім'я ПРІЗВИЩЕ)

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Дніпро – 2022 рік

2

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

Faculty «Computer technologies and systems»
Department «Computer information technology»

Explanatory Note
to Bachelor's Thesis

on the topic: «Comb filtering of time series»
according to educational curriculum « Software engineering »
in the Speciality: «121 Software engineering»

Done by the student of the group ПІ31812: /Volodymyr POHREBNIAK/
Scientific Supervisor: /Viktor SHYNKARENKO/
Normative controller: /Olena KUROPIATNYK/

Dnipro – 2022

3

Міністерство освіти і науки України

Український державний університет науки і технологій

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

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

Дата _____

З А В Д А Н Н Я

на кваліфікаційну роботу бакалавра
студенту Погребняку Володимиру Миколайовичу
1. Тема роботи: «Гребінкова фільтрація часових рядів»
Керівник роботи: Шинкаренко Віктор Іванович
затверджені наказом № 77 ст від 08.12.2021
2. Строк подання студентом роботи: 14.06.2022 р.
3. Вихідні дані до роботи: _____

4. Зміст пояснювальної записки (перелік питань, які потрібно опрацювати):
4.1 Аналітична частина: збір та аналіз вимог, опис аналогів.

4.2 Основна частина: проєктування, розробка програми, тестування

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):титулка, зміст, вступ, завдання, огляд програми, висновки.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі	31.01.2022 – 05.01.2022	
2	Огляд літератури та аналіз аналогів	16.01.2022 – 24.01.2022	
3	Розробка структур вхідних і вихідних даних	25.01.2022 – 02.02.2022	
4	Визначення вимог до програми. Вибір та обгрунтування мови програмування	03.02.2022 – 10.02.2022	
5	Узгодження та затвердження ТЗ	11.02.2022 – 18.02.2022	30%
6	Розробка та програмування логіки програми	19.02.2022 – 01.03.2022	
7	Розробка і реалізація інтерфейсу користувача	02.03.2022 – 20.03.2022	
8	Відлагодження програми	21.03.2022 – 24.03.2022	60%
9	Розробка, узгодження та затвердження програмної документації	25.03.2022 – 03.04.2022	
10	Подання кваліфікаційної роботи до кафедри	04.04.2022– 12.06.2022	100%
11	Захист кваліфікаційної роботи на засіданні Екзаменаційної комісії	22.06.2022	

Студент _____ (підпис) Володимир ПОГРЕБНЯК (Ім'я ПРІЗВИЩЕ)

Керівник роботи _____ (підпис) Віктор ШИНКАРЕНКО (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка складається з 5 розділів:

– вступ – в даному розділі описується сутність розробки, її актуальність. Складається з 1 сторінки;

– збір вимог до програмного забезпечення – у цьому розділи описуються аналоги програми та література по даній предметній області, а також проводиться опит зацікавлених сторін для формування найбільш повних вимог до програмного забезпечення. Складається з 11 сторінок;

– зовнішнє і внутрішнє проектування – у цьому розділі проведений огляд вхідних і вихідних даних, формалізація задачі, розробка фізичного проекту, приводиться опис об’єктно-орієнтованого проектування, проектування інтерфейсу користувача, ескізи форм, аналіз проекту, проектування динаміки системи, вибір мови програмування. Складається з 2 сторінок;

– тестування та налагодження – включає в себе вибір стратегії тестування, опис тестів методом «білого» ящика. Також аналіз помилок їх вплив на систему та вирішення проблеми. Складається з 2 сторінок;

– висновки. Складається з 1 сторінки;

– список літератури – включає в себе бібліографічний список використаної літератури. Складає 1 сторінки;

– додатки – містить технічне завдання і робочий проект.

Кількість таблиць: 9 шт. Кількість рисунків: 10 шт.

Ключові слова: часовий ряд, фільтрація, гребінкова фільтрація, ряд, функція, число.

ЗМІСТ

ВСТУП

Актуальність дослідження: робота є актуальною, тому що часові ряди використовуються майже усюди. У багатьох приладах або програмах для моніторингу наявні часові ряди. Але виникає такий момент коли потрібен не весь ряд, а лише за певний час або ж навіть окремі точки. Саме для цього потрібне таке забезпечення, а саме у даній ситуації гребінкова фільтрація.

Мета: теоретично дослідити наукову літературу з питань гребінчастої фільтрації часових рядів та підкріпити це емпіричним дослідженням, у ході якого буде створений програмний додаток.

Експлуатаційне призначення: за допомогою даного додатку користувачі мають змогу використовувати у різних цілях для побудови більш конкретних графіків.

1. ЗБІР ТА АНАЛІЗ ВИМОГ

У біомедичній обробці сигналів та обробці сигналів загалом підходи гребінчастої фільтрації представляють важливий клас фільтрів, які відіграють важливу роль у різних галузях, таких як виділення або видалення періодичних сигналів і гармонійних компонентів, обробка мовлення та звукових сигналів, процеси децимації, передбачення і оцінка геофізичних сигналів, і відхилення лінії електропередач [1-5]. У найпростішій формі гребінчастий фільтр можна розглядати як комбінацію режекторних фільтрів, у яких нульові частоти періодично зустрічаються по всій смузі фільтра. Іншим дуже популярним підходом до гребінчастої фільтрації є звичайний фільтр КІХ ковзного середнього, зазначений у

$$yn=1M\sum k=0M-1xn-k, \tag{1.1}$$

представленні яких у z -області та дискретній часовій реалізації показано відповідно в (1.2) і рис. 1.1.

$$Y(z)X(z)=1M(1-z-M)(1-z-1)=HMAF(z), \tag{1.2}$$

з $M = f_s / f_M$, де f_s - частота дискретизації, а f_M - основна періодична нульова частота.

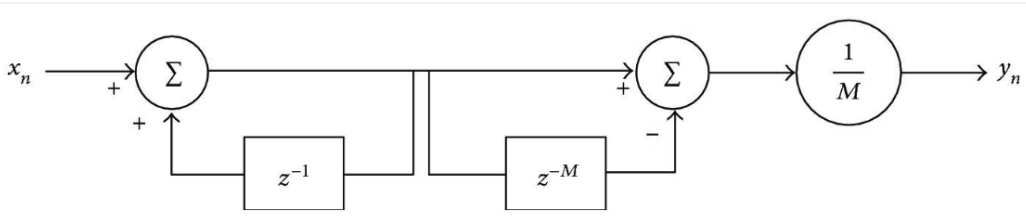

Рисунок 1. 1. – Реалізація ковзного середнього за дискретним часом, описана в (1.2).

Реалізація гребінчастого фільтра, зазначена в (1.2), широко використовується через свою обчислювальну ефективність. Однак, як обмеження, (1.2) забезпечує амплітудну характеристику з низьким загасанням у смугах затримки фільтра, а також нерівномірним посиленням і високим загасанням у смугах пропускання, як показано на рис. 1.1. Крім того, незважаючи на частичну лінійність фазової характеристики на

рис. 1.2. це може спровокувати збільшення фазової затримки при більш високих значеннях M [1 , 4].

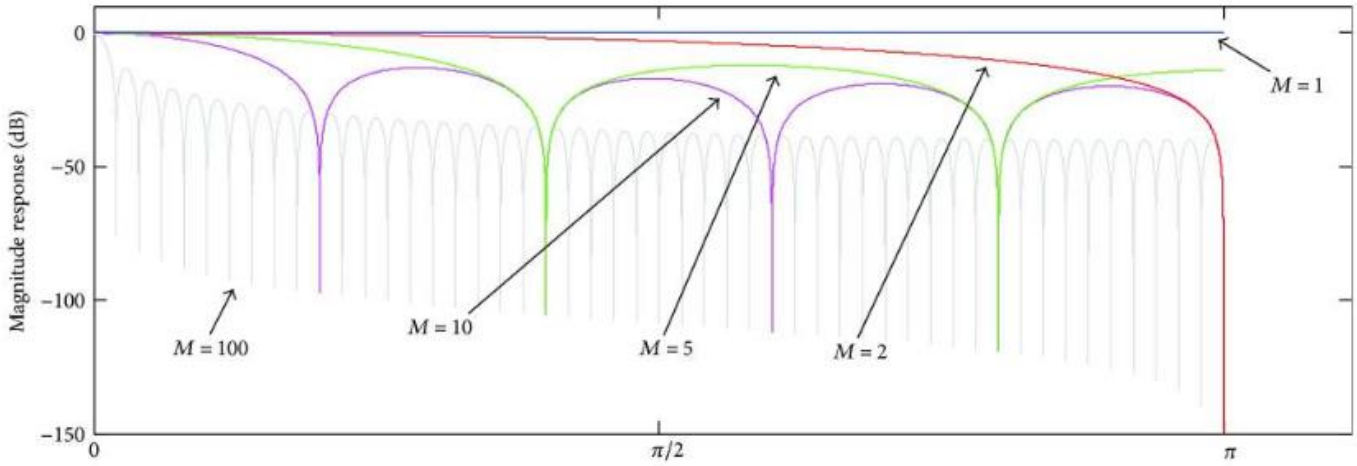

Рисунок 1.2. – Нерівномірне посилення та згасання у смугах пропускання

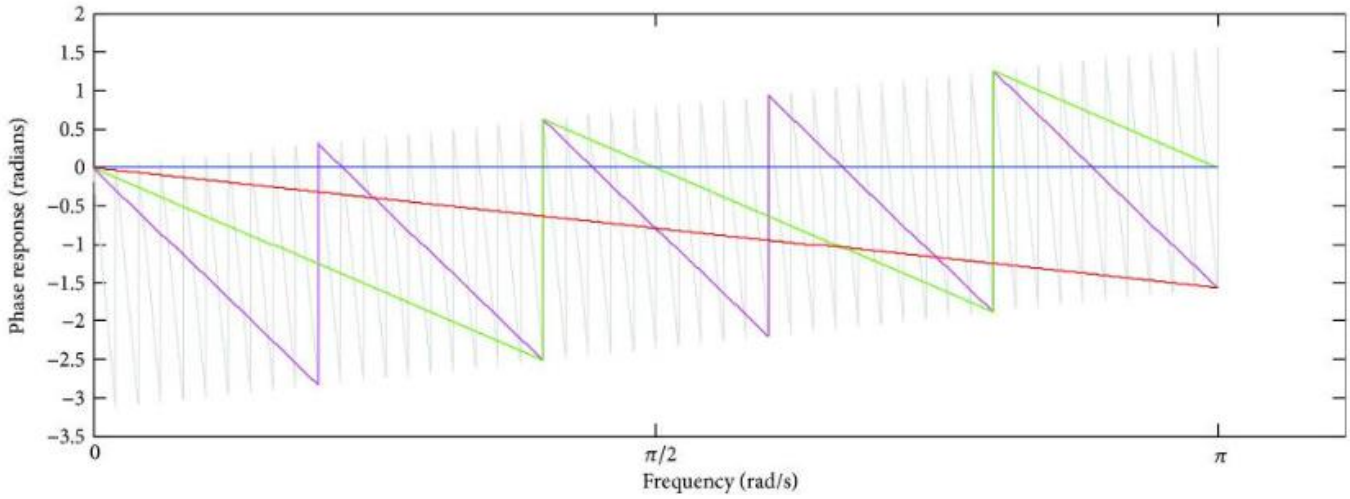

Рисунок 1.3. – Фазова затримка

Такі характеристики небажані в деяких програмах і далекі від характеристик ідеального гребінчастого фільтра: нульовий підсилення на частотах вирізів, рівномірний і одиничний підсилення в діапазонах пропускання і відсутність впливу на фазу сигналу. Щоб зробити реалізацію гребінчастого фільтра (1.2) більш селективною або наближеною до ідеальної поведінки, в літературі були запропоновані деякі стратегії. Наприклад, цього можна досягти введенням полюсів у передатну функцію (1.2), як зазначено в [4, 6, 7].

$$MOD(z)=1−z−M1−z−11−r\cdot z−11−(r\cdot z−1)M \qquad (1.3)$$

де значення параметра r міститься в інтервалі $0 \leq r < 1$. Як зазначають Проакіс і Манолакіс [4], вставка полюсів у (1.2) має ефект введення резонансу в околиці нуля, тим самим провокуючи зменшення пропускної здатності смуг затримання. Таким чином, нулі $z = e^{j 2 \pi k / M}$, розміщені на одиничному колі в z-площині, матимуть поблизу полюса $z = r \cdot e^{j 2 \pi k / M}$. Гребінчастий фільтр, описаний у (1.3) успішно використовується в ряді застосувань, наприклад у гармонічних компенсаторах і випрямлячах в енергосистемах [6 , 7]. Однак одним із обмежень цього підходу є зменшення загасання в смугах затримки, оскільки значення r зростає до 1. Крім того, існує компроміс між значеннями M та r , що залежить від вимоги до продуктивності фільтр: з одного боку, використання вищих значень M робить цей метод дорогим для обчислень щодо використання пам'яті. З іншого боку, експоненціальне зменшення потужності r^M дозволяє реалізувати фільтр за допомогою комп’ютерного блоку з нижчою роздільною здатністю. Таким чином, існує компроміс між значенням M і роздільною здатністю комп’ютерної одиниці [6].

Іншою запропонованою стратегією для підвищення селективності гребінчастої фільтрації, що забезпечується (1.2), є підхід усереднення у часовій області. Усереднення у часовій області складається з свого роду гребінчастого підходу, заснованого на когерентному процесі виявлення, при якому оцінка та усунення періодичної активності здійснюються шляхом усереднення повторюваних послідовностей періодичного сигналу $p(t)$, що спостерігається у вхідному сигналі, $x(m)$.

$$x(t) = p(t) + e(t) \tag{1.4}$$

У (1.4) $e(t)$ являє собою неперіодичну складову $x(t)$, яка може бути шумовим сигналом або деяким випадковим процесом. За припущення, що $p(t)$ і $e(t)$ некорельовані, підсумовування N наступних сегментів $x(t_i)$, відповідних періодичному сигналу, призводить до когерентного підсумовування $p(t)$ [8].

Усереднення у часовій області — це добре встановлений підхід гребінчастої фільтрації, який широко використовується для оцінки та виділення періодичних сигналів, що зустрічаються в явищах, які включають деякі обертові механізми [2 , 8 , 9]. Також були запропоновані підходи на основі усереднення у часовій області для оцінки та усунення градієнтного артефакту з сигналу ЕКГ, наприклад метод віднімання середнього артефакту [10 , 11].

Артефакт градієнта складається з періодичної інтерференції напруги форми хвилі, яка індукується в електричному потенціалі, записаному в шкірі голови людини (потенціал шкіри голови), за допомогою швидко змінюваних градієнтів магнітного поля та радіочастотних імпульсів, які використовуються в послідовностях МРТ під час одночасного отримання даних ЕКГ та фМРТ [12 , 13]. Однак одним з обмежень усереднення у часовій області є його висока залежність від точної вибірки періодичної форми сигналу $p(t)$. Виникнення похибок тремтіння може призвести до неточної вибірки сигналів усереднення, що може погіршити ефективність методу. Таким чином, період повторюваної форми сигналу повинен бути точним кратним інтервалу дискретизації. Паралельно, період $p(t)$ повинен бути точно відомий, для чого потрібен зовнішній тригер або опорний сигнал, що надається додатковим обладнанням [8 , 14]. У разі придушення градієнтного артефакту від сигналу ЕКГ рух суб’єкта або невеликі дрейфи також можуть поставити під загрозу роботу алгоритму, оскільки вони змінюють морфологію та форму артефакту таким чином, що неможливо отримати точна оцінка. Рухи суб’єкта або невеликі дрейфи також провокують розширення спектральних ліній гармонійного артефакту [15], ослаблення якого може не враховуватися гребінчастим фільтром усереднення у часовій області. Як наслідок, залишкові артефакти залишаються в коригованій ЕКГ після віднімання розрахункової періодичної форми хвилі.

Низка біомедичних та інших застосувань обробки сигналів вимагають використання підходів гребінчастої фільтрації, які виконують видалення або вилучення періодичних сигналів з високим ступенем селективності. Що стосується усунення періодичних сигналів, то гребінчастий фільтр повинен бути здатним придушувати гармоніки, пов’язані з періодичним сигналом, і одночасно зберігати

стохастичний компонент або шумовий сигнал відповідно до рівня якості, який вимагає додаток. Таким чином, важливо переконатися, що ефективність гребінчастого підходу фільтрації відповідає вимогам до селективності програми. Оскільки існуючі методи не завжди відповідають таким вимогам, у літературі часто описуються дослідження та пропозиції нових підходів до підвищення селективності гребінчастої фільтрації.

Висновки до розділу 1

Тема є актуальною через багаточисленні дослідження. Так як часові ряди використовуються майже усюди, то їх фільтрація для подальшого аналізу також потрібна. Ця тема досліджується у багатьох галузях та буде актуальною ще дуже довго.

2. ОПИС АНАЛОГІВ

Насправді є дуже багато аналогів у цій сфері. Наприклад часто фільтрацію часових рядів використовують на різноманітних валютних біржах. Наприклад криптовалютна біржа Binance.

Рисунок 2.1. Часовий ряд на Binance.

Також можна знайти часові ряди навіть у прогнозах погоди. Адже погода також потребує аналізу даних температур та інших показників. Ось маємо приклад на рис. 2.2 де можемо помітити використання часових рядів та їх фільтрацію наприклад на тиждень:

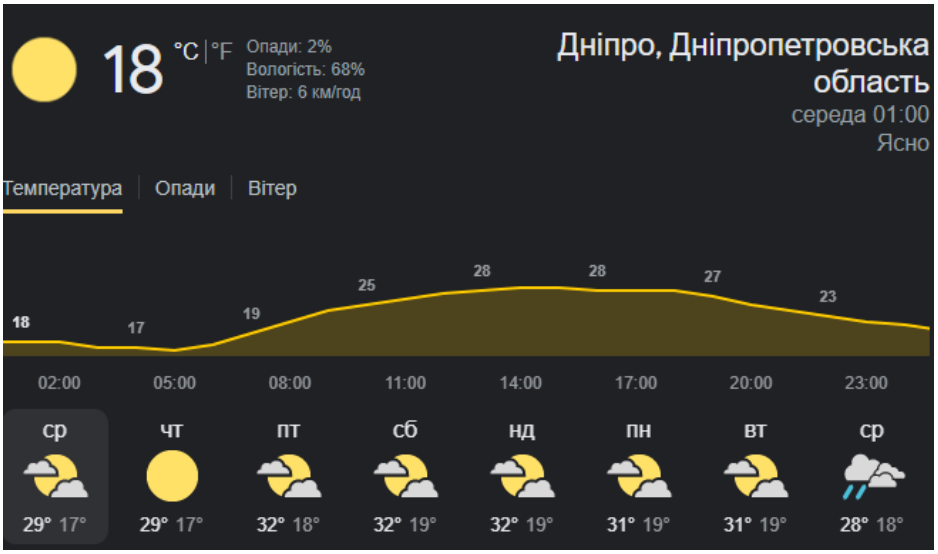

Рисунок 2.2. Часовий ряд у прогнозі погоди.

Навіть у кишеньковому гаджеті може знайтися така функція, для цього просто потрібно зайти до налаштувань:

Рисунок 2.3. У налаштуваннях телефону також можна знайти часовий ряд.

Проходячи комісію у лікарні ми також зустрічаємося з часовими рядами. Наприклад електрокардіограма:

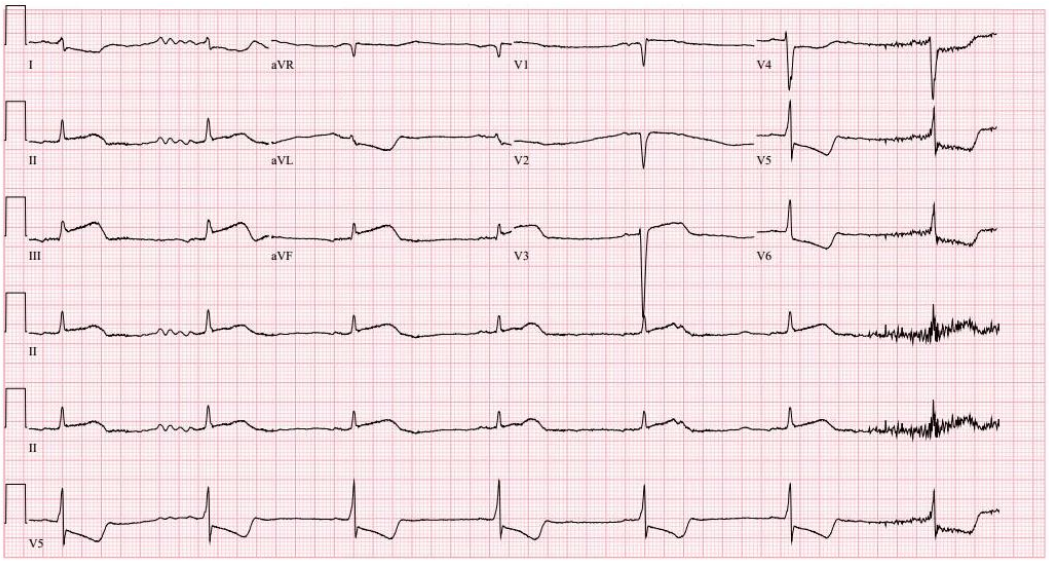

Рисунок 2.4. Електрокардіограма також без часового ряду не існувала б.

Але існує така проблема, що потрібно відфільтрувати з певним кроком деякі елементи. Та більша кількість таких додатків, програмних продуктів не дуже з такою задачею справляються. Тож мною було вирішено створити додаток для таких цілей.

Висновки до розділу 2

Оглянувши аналоги можна підкріпити слова про різносторонні використання часових рядів. Від різних бірж по продажу, до важливого для нашого здоров’я напрямку – медицини. Отже, тема дійсно є актуальною та потребує подальшого вивчення.

3. ПРОЄКТУВАННЯ

3.1. Зовнішнє проєктування

Визначимо зовнішні взаємодії майбутнього програмного продукту із зовнішнім середовищем без конкретизації його внутрішньої структури. Для цього розкриємо такі питання як: функціональне призначення, експлуатаційне призначення, функціональні вимоги, вхідні та вихідні дані, опис зовнішнього інформаційного середовища.

Функціональне призначення: гребінкова фільтрація часових рядів.

Експлуатаційне призначення: зменшення часових витрат на перебудову відфільтрованого часового ряду.

Функціональні вимоги: Імпортування даних з файлу типу «*.xlsx», вибір масштабування графіків, вибір кроку фільтрації ряду та вибір точок які потрібно відфільтрувати.

Вхідні дані: файл формату «*.xlsx». У листі документу EXCEL у першому стовпці (A) знаходяться дати, а у другому стовпці (B) знаходитимуться значення відповідні до цих дат.

Вихідні дані: на екран повинні виводитись часові ряди «До фільтрації» та «Після фільтрації». А також вивід відфільтрованого часового ряду до нового файлу ексель.

3.2. Внутрішнє проєктування

Основна частина додатку знаходиться з лівої сторони. Роботу з додатком потрібно починати з верхньої частини. До кожної кнопки прив’язані свої властивості. Сценарій являється таким: спочатку імпортується файл, потім користувач обирає фільтрацію, розмір кроку та точки які потрібно прибрати, натиснути на кнопку «Фільтр» і згодом можна зберегти файл на диск D.

3.2.1. Проєктування інтерфейсу користувача

Інтерфейс будується на основі екранних форм. Тож на екрані з’явиться вікно, у якому можна буде вже працювати.

А це означає що потрібно визначитись зі складом інформації, яка буде виводитись на екран користувача.

Отже, у програмі потрібно бачити зміни, а це значить, що там будуть часовий ряд до фільтрації та після неї. Щоб додати графік до фільтрації, нам потрібна кнопка, яка буде відповідати за імпорт даних для часового ряду. А також потрібно визначити розмір кроку для фільтрації, для цього є випадаючий список, де в залежності від кількості елементів можна буде вибрати розмір кроку. Також потрібно вибирати які саме елементи по порядку будуть видалятись, а отже потрібно мати список, який можна відмічати галочками, щоб їх видалити.

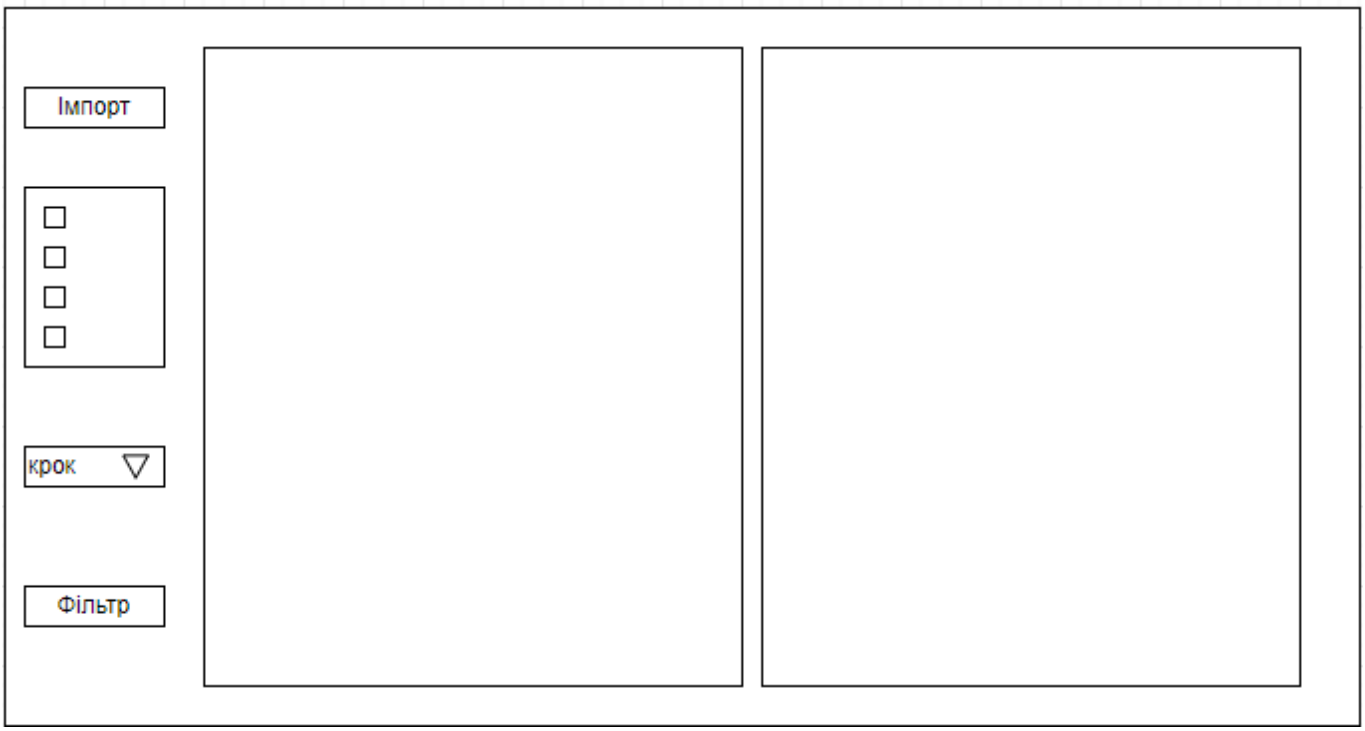

Рисунок 3.1 Макет розміщення даних на екрані.

Після макету інтерфейсу користувача слід розглянути діаграму діяльності:

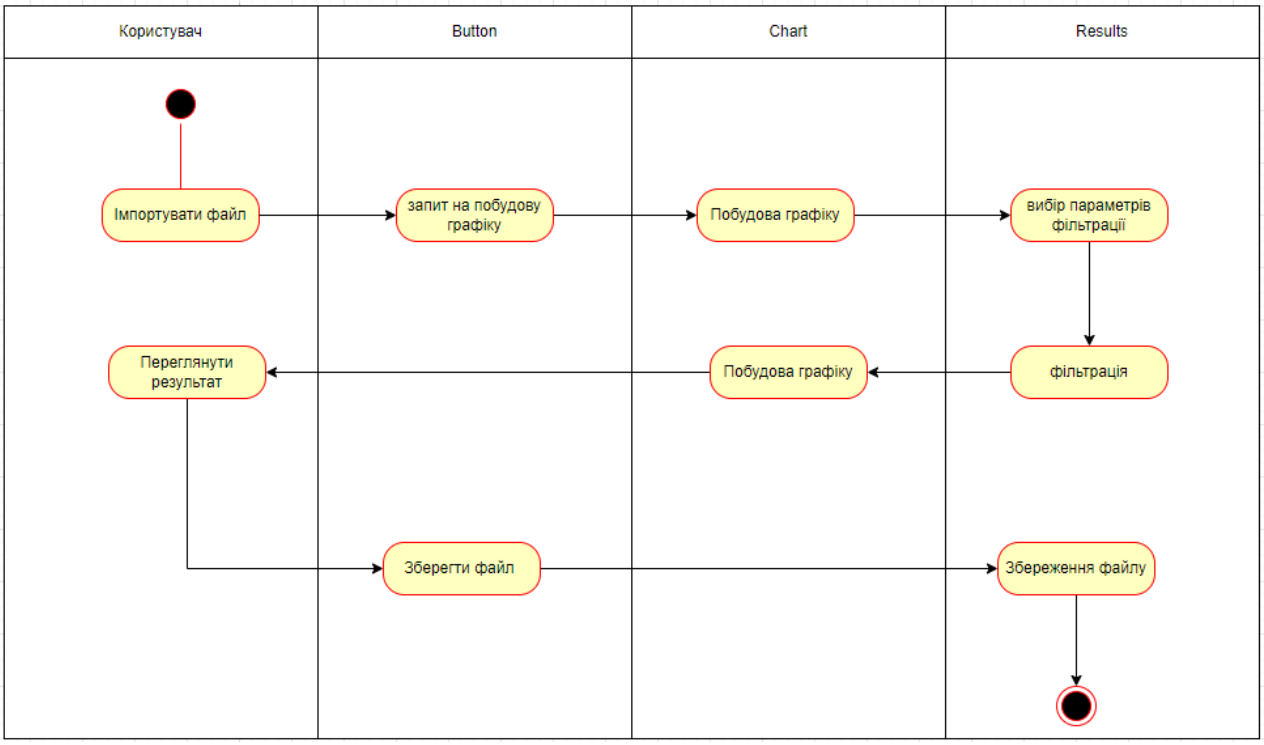

Рисунок 3.2. Діаграма діяльності

Висновки до розділу 3

Склавши макет вікна програмного додатку та діаграму діяльності, можна визначитись з мовою та середовищем розробки, також за діаграмою діяльності можна розробити алгоритми.

4. РОЗРОБКА ПРОГРАМИ

На основі попереднього матеріалу було визначено мову програмування, а саме C# у середовищі Visual Studio 2022.

Спочатку потрібно розробити алгоритми. Так як програма починається з імпортування файлу, то і алгоритм буде побудований для цього перший.

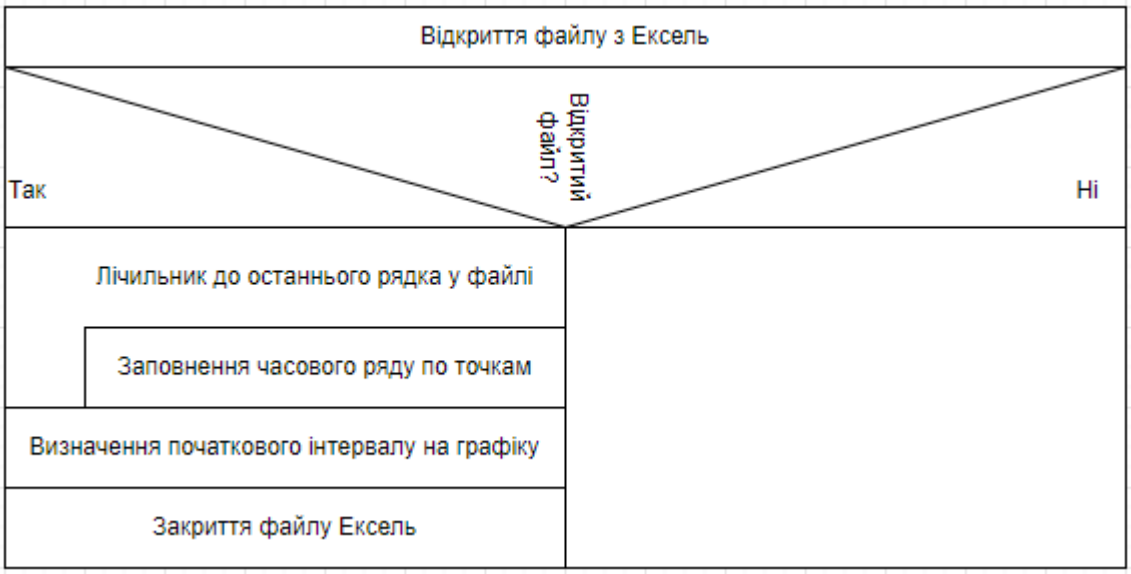

Рисунок 4.1. Алгоритм імпорту файлу

Маємо перший алгоритм. Далі нам потрібно виконувати фільтрування.

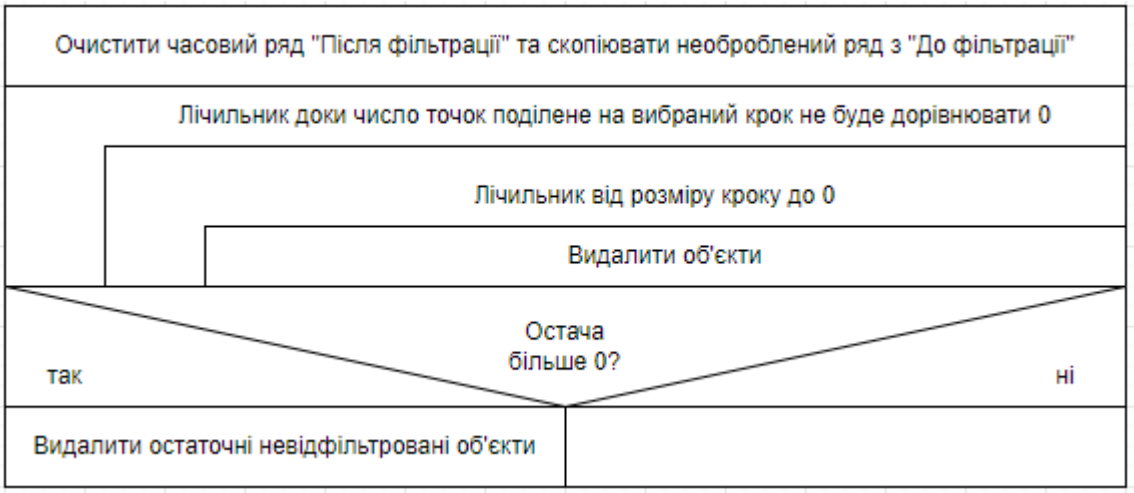

Рисунок 4.2. Алгоритм фільтрації

Ще один алгоритм, але вже для фільтрації, де ми можемо спостерігати яким чином буде відтворюватись процес.

Залишився ще один алгоритм, це алгоритм експорту відфільтрованих даних до

таблиці EXCEL.

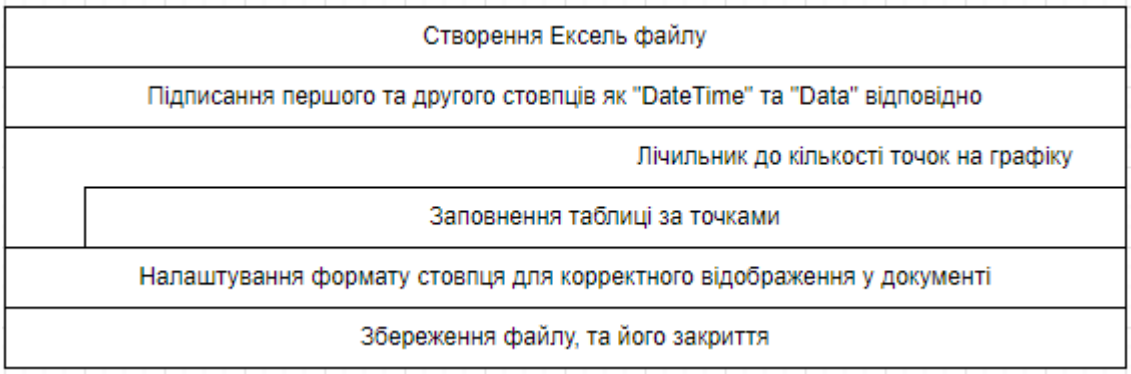

Рисунок 4.3. Алгоритм збереження часового ряду

Висновки до розділу 4

Використовуючи діаграму діяльності та теорію було створено алгоритми програмного додатку. За цими алгоритмами можна далі розробити програмний додаток.

5. ТЕСТУВАННЯ

Тест – набір вхідних і вихідних даних для разового виконання програми, що задовольняє специфікації. Тестування полягає у динамічній перевірці поведінки програми на скінченній множині тестових даних, вибраних спеціальним чином з нескінченного вхідного простору, на відповідність встановленій очікуваній поведінці. Методи тестування різняться підходами до проєктування тестів. Традиційно методи функціонального ручного тестування розділяють на дві категорії – чорна скринька (без доступу до тексту програми) та біла скринька (з доступом).

5.1. Тестування білою скринькою

Метод покриття операторів:

Таблиця 5.1 – Покриття умов

Тест/умова	if (ofd.ShowDialog() != DialogResult.OK)		if (comboBox2.SelectedIndex == 0 || comboBox2.SelectedIndex == -1)		if (comboBox2.SelectedIndex == 1)	
	+	-	+	-	+	-
1	*		*	*	*	*
2	*		*	*	*	*
3	*		*	*	*	*

Тест/умова	if (checkedListBox1.GetItemChecked(j) == true)		if (nume > 0)	
	+	-	+	-
1	*	*	*	*
2	*	*	*	*
3	*	*	*	*

Таблиця 5.2. – Покриття напряму

Тест/напрям	ofd.ShowDialog != DialogResult.OK		comboBox2.SelectedIndex == 0 || comboBox2.SelectedIndex == -1		comboBox2.SelectedIndex == 1	
	+	-	+	-	+	-
1	*		*	*	*	*
2	*		*	*	*	*
3	*		*	*	*	*

Тест/напрям	checkedListBox1. GetItemChecked(j) == true	
	+	-
1	*	*
2	*	*
3	*	*

Покриття операторів: Ofd, xlApp, xlWB, xlSht, i, j, iLastRow, numPoints, list, xlWorkBook, xlWorkSheet, formatRange, num, nume.

АНАЛІЗ ТА ВИСНОВКИ

Виконуючи цю роботу я отримав практичні навички у пошуку інформації, та дізнався про нове. А також мені чомусь не здавалося важливим питання у галузі часових рядів, бо не поглибившись у цю тему, я мав підозри, що ця тема ніде не використовується, а виявилось навпаки. Отже, було створено додаток, який в змозі фільтрувати потрібні часові ряди. Виконуючи цю роботу я побачив багато аналогів, де використовується фільтрація часових рядів. Було цікаво читати літературу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rangayyan R. M. Biomedical Signal Analysis
2. S Braun. The synchronous (time domain) average revisited
3. J Meher, PK Meher, G Dash. Improved comb filter based approach for effective prediction of protein coding regions in DNA sequences
4. Proakis J. G., Manolakis D. G. Digital Signal Processing: Principles, Algorithms, and Applications.
5. Zeng Z., Xie Y., Wang Y., Guan Y., Zhang X, Li L. An improved harmonic current detection method based on parallel active power filter
6. Tahir M. Improving dynamic response of active harmonic compensator using digital comb filter
7. Prodić A., Maksimović D., Chen J., Erickson R. W. Self-tuning digitally controlled low-harmonic rectifier having fast dynamic response
8. S Braun. The extraction of periodic waveforms by time domain averaging
9. P D McFadden. A revised model for the extraction of periodic waveforms by time domain averaging
10. P J Allen, O Josephs, R Turner. A method for removing imaging artifact from continuous EEG recorded during functional MRI
11. Moosmann M., Ritter P., Becker R.,Villringer. A. Visual evoked potentials recovered from fMRI scan periods
12. Mori T., Anami K., Tanaka F., et al. Stepping stone sampling for retrieving artifact-free electroencephalogram during functional magnetic resonance imaging
13. WX Yan, KJ Mullinger, MJ Brookes, R Bowtell. Understanding gradient artefacts in simultaneous EEG/fMRI
14. H Mandelkow, P Halder, P Boesiger, D Brandeis. Synchronization facilitates removal of MRI artefacts from concurrent EEG recordings and increases usable bandwidth
15. G S Spencer. EEG-fMRI: novel methods for gradient artefact correction

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Перший проректор Українського
державного
університету науки і технологій
Анатолій РАДКЕВИЧ
18.02.22

ПРОГРАМА
ГРЕБІШКОВА ФІЛЬТРАЦІЯ

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01247-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
18.02.22

Керівник розробки
Віктор ШИНКАРЕНКО
18.02.22

Виконавець
Володимир ПОГРЕБНЯК
18.02.22

Нормоконтролер
Олена КУРОП'ЯТНИК
18.02.22

2022

ЗАТВЕРДЖЕНО
44165850. 01247-01-ЛЗ

ПРОГРАМА
ГРЕБІШКОВА ФІЛЬТРАЦІЯ

Технічне завдання

44165850.01247-01

Листів 14

2022

АНОТАЦІЯ

Документ 44165850.01247-01 «Гребінкова фільтрація. Технічне завдання» входить до складу програмної документації на

Програмний додаток, що надає можливість користувачам виконати фільтрацію часового ряду методом гребінця.

У даному документі представлено технічне завдання до програмного додатку.

ЗМІСТ

Вступ

Програма «Гребінкова фільтрація», що надає можливість користувачам відфільтрувати потрібні їм значення у часовому ряду.

Часто буває, що потрібно відфільтрувати дані за деякі дати, та на поміч цьому приходять програми що дозволяють це зробити.

Даний програмний продукт забезпечить зручністю та швидкістю фільтрування потрібних даних.

Область застосувань: програмний продукт призначений не тільки для індивідуального використання, його можуть використовувати і компанії, які цього потребують.

1. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ ректора Українського державного університету науки і технологій про призначення керівників та затвердження тем бакалаврських робіт №77-ст від 08.12.21. Тема роботи «Гребінкова фільтрація часових рядів». Науковий керівник – проф. Шинкаренко В.І.

2. ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення – фільтрація часових рядів за запитами користувача.

Експлуатаційне призначення – зменшення витрат ресурсів на попередню обробку часових рядів.

3. ВИМОГИ ДО ДОДАТКУ

3.1 Вимоги до функціональних характеристик

Програма провинна забезпечити зручність користуванням для фільтрування потрібних даних у часовому ряді.

Вхідні дані:

- дати та значення для побудови графіку, дати які потрібно відфільтрувати.
- дані вводяться з текстового файлу або з файлу таблиць формату EXEL або вручну за клавіатури та миші.

Вихідні дані: часовий ряд, та його відфільтрована версія, екпортований файл ексель.

3.2. Вимоги до надійності

Вимоги до надійності наступні:

- для полів ведення необхідно забезпечити контроль ведених даних;
- при оновленні даних забезпечити показ відповідних повідомлень про стан роботи програми та результати виконання операцій;
- наявність архівної копії тексту програми на зовнішньому носії;
- не допускається наявність логічних помилок у програмі

3.3 Умови експлуатації

Програмний продукт повинен використовуватись у приміщеннях, які відповідають умовам роботи ЕОМ, а саме мають такі кліматичні, санітарні та гігієнічні умови, які відповідають ДНАОП 0.00-1.31-99 (див. табл. 1).

Таблиця 3.1. – Кліматичні умови

Пора року	Категорія робіт згідно з ГОСТ 12.1-005-88	Температура повітря, град.С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
		оптимальна	оптимальна	оптимальна
Холодна	легка-1 а	22 - 24	40 - 60	0,1
	легка-1 б	21 - 23	40 - 60	0,1
Тепла	легка-1 а	23 - 25	40 - 60	0,1
	легка-1 б	22 - 24	40 - 60	0,2

Працювати з програмою може людина, що має навички роботи з персональним комп’ютером та ознайомлена з керівництвом користувача.

3.4. Вимоги до складу та параметрів технічних засобів

Основні вимоги висуваються тими програмами, які будуть використовувати фільтрацію часових рядів. Саме для фільтрації потрібна пам’ять достатня для розміщення часових рядів. Для самої програми фільтрації достатньо 1МБ.

3.5. Вимоги до інформаційної та програмної сумісності

Програмний продукт розробляється для всіх видів операційних систем сімейства Windows. Середовище розробки: Visual Studio 2022.

3.6 Вимоги до маркування і упаковки

Упаковка програмного додатку, включаючи документацію повинна бути захищена від пошкоджень різного роду (механічних, кліматичних). На упаковці повинно бути вказана назва продукту, номер версії (якщо вона змінювалась), мінімальні системні вимоги. На зворотній стороні упаковки вказується розробник та його юридична адреса (рис. 3.1).

Програмний додаток «ГРЕБІШКОВА ФІЛЬТРАЦІЯ»	Програмний додаток «ГРЕБІШКОВА ФІЛЬТРАЦІЯ» Розробник: Погребняк В. М. Кафедра «КІТ», УДУНТ м. Дніпро, вул. Лазаряна 2 2022

Рисунок 3.1. – Маркування

3.7 Вимоги до транспортування і зберігання

Програмний виріб міститься у запакованому архіві ГРЕБІШКОВА_ФІЛЬТРАЦІЯ.zip. Копія додатку розміщується на USB носії.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації мають входити:

- специфікація;
- текст програми;
- опис програми;
- керівництво користувача. Керівництво з фільтрації часових рядів.

Вся документація до програмного додатку повинна задовольняти вимоги до програмної документації.

5 ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Фінансування на розробку програмного забезпечення відсутнє. Прибуток за застосування не передбачається. Техніко-економічні показники не потребують розрахунку.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

В табл. 6.1 приведені стадії та етапи розробки програмного продукту.

Таблиця 6.1. – Стадії та етапи розробки

Стадія	Зміст	Строки виконання
Технічне завдання	Постановка задачі, збір інформації, виріб та обґрунтування критеріїв розробки. Попередній вибір методів рішення задач. Визначення вимог до технічних засобів. Узгодження і затвердження технічного завдання.	31.01.22 - 18.02.22
Робочий проект	Програмування та налагодження програми.	19.02.22 - 01.05.22
	Тестування програми	02.05.22 - 24.05.22
	Розробка, узгодження і затвердження програмної документації.	25.05.22 - 12.06.22

7 ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Контроль за виконанням роботи здійснює керівник розробки.

Прийом програми здійснює комісія у складі, який визначає університет.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Додаток Б

ЗАТВЕРДЖУЮ
Перший проректор Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ
10.06.22

ПРОГРАМА
ГРЕБІШКОВА ФІЛЬТРАЦІЯ

Робочий проект
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01247-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
10.06.22

Керівник розробки
Віктор ШИНКАРЕНКО
10.06.22

Виконавець
Володимир ПОГРЕБНЯК
10.06.22

Нормоконтролер
Олена КУРОП'ЯТНИК
10.06.22

2022

Позначення	Найменування	Примітки
	Документація	
44165850. 01247-01-ЛЗ	Лист затвердження	
44165850. 01247-01 12 01	Текст програми	
44165850. 01247-01 13 01	Опис програми	

Додаток В

ЗАТВЕРДЖЕНО
44165850.01247-01 12 01-ЛЗ

ПРОГРАМА
ГРЕБІШКОВА ФІЛЬТРАЦІЯ

Текст програми

44165850.01247-01 12 01

Листів 6

2022

АНОТАЦІЯ

Документ 44165850.94208-01 12 01 «Програма Гребішкова фільтрація. Текст Програми» входить до складу програмної документації на програму, що надає можливість користувачам фільтрувати часові ряди.

У даному документі представлений текст програм. Програми написані на мовах C#. Об’єм пам’яті, що необхідний для виконання програми, складає 0.5 ГБ. Програма функціонує в середовищі Windows.

ЗМІСТ

1. ОПИС МОДУЛІВ

1.1. Модуль Button_import

– Необхідний для реалізації імпорту файлу EXCEL до програми. Також відповідає за виведення часового ряду «До фільтрації».

1.2. Модуль Button_filter

– Відповідає за фільтрацію часового ряду та виводить його на екран.

1.3. Модуль Button_export

– Необхідний для експорту файлу формату «*.xlsx» у вигляді таблиці

2. ТЕКСТ ПРОГРАМИ

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Excel = Microsoft.Office.Interop.Excel;
using System.Runtime.InteropServices;



namespace WindowsFormsApp1
{

    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
```

```
    }

    void Button_import(object sender, EventArgs e)
    {
        chart1.Series["До фільтрації"].Points.Clear();

        OpenFileDialog ofd = new OpenFileDialog();
        ofd.DefaultExt = "*.xls;*.xlsx";
        ofd.Filter = "Microsoft Excel (*.xls*)|*.xls*";
        ofd.Title = "Выберите документ";
        if (ofd.ShowDialog() != DialogResult.OK)
        {
            MessageBox.Show("Вы не выбрали файл", "Загрузка данных...",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }


        Excel.Application xlApp = new Excel.Application(); //Excel
        Excel.Workbook xlWB; //рабочая книга
        Excel.Worksheet xlSht; //лист Excel
        xlWB = xlApp.Workbooks.Open(ofd.FileName); //название файла Excel
        xlSht = xlWB.Worksheets[1]; //название листа или 1-й лист в книге xlSht =
xlWB.Worksheets[1];

        string i;
        string j;
```

```
            int iLastRow = xlSht.Cells[xlSht.Rows.Count,
"A"].End[Excel.XlDirection.xlUp].Row;


            for (int q = 1; q <= iLastRow; q++)
            {

                i = xlSht.Cells[q, 1].Value.ToString("dd-MM-yyyy");
                j = xlSht.Cells[q, 2].Value.ToString();
                chart1.Series["До фільтрації"].Points.AddXY(i, j);
            }
            chart1.ChartAreas[0].AxisX.Interval = 1;

            xlWB.Close(false); //закрываем книгу, изменения не сохраняем
            xlApp.Quit(); //закрываем Excel
            copychart();
            releaseObject(xlSht);
            releaseObject(xlWB);
            releaseObject(xlApp);
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            chart1.ChartAreas[0].AxisX.Interval = comboBox1.SelectedIndex + 1;

        }

        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (comboBox2.SelectedIndex == 0 || comboBox2.SelectedIndex == -1)
```

```
    {
        copychart();
    }

    if (comboBox2.SelectedIndex == 1)
    {
        int numPoints = chart1.Series[0].Points.Count;
        comboBox4.Items.Clear();
        for (int i = 0; i < numPoints; i++)
        {
            comboBox4.Items.Add(i);
        }
    }
}

private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
{
    chart2.ChartAreas[0].AxisX.Interval = comboBox3.SelectedIndex + 1;
}

private void checkedListBox1_SelectedIndexChanged(object sender, EventArgs e)
{

}

private void Button_filter(object sender, EventArgs e)
{
    for (int i = 0; i < checkedListBox1.Items.Count; i++)
    {
        checkedListBox1.SetItemChecked(i, false);
```

```
    }
}

private void comboBox4_SelectedIndexChanged(object sender, EventArgs e)
{
    int list = comboBox4.SelectedIndex;
    int numPoints = chart2.Series[0].Points.Count;

    checkedListBox1.Items.Clear();
    for (int i = 1; i <= list; i++)
    {
        checkedListBox1.Items.Add(i);

    }
}

private void Button_export(object sender, EventArgs e)
{
    checkedclear();
}

private void button4_Click(object sender, EventArgs e)
{

    Excel.Application xlApp;
    Excel.Workbook xlWorkBook;
    Excel.Worksheet xlWorkSheet;

    object misValue = System.Reflection.Missing.Value;
```

```
xlApp = new Excel.Application();
xlWorkBook = xlApp.Workbooks.Add(misValue);
xlWorkSheet = (Excel.Worksheet)xlWorkBook.Worksheets.get_Item(1);

    xlWorkSheet.Cells[1, 1] = "DateTime";
    xlWorkSheet.Cells[1, 2] = "Data";

    for (int j = 0; j < chart2.Series[0].Points.Count; j++)
    {
    string i = chart2.Series[0].Points[j].AxisLabel[0].ToString();
        for (int c=1;c<10;c++)
        {
            i += chart2.Series[0].Points[j].AxisLabel[c].ToString();
        }

        xlWorkSheet.Cells[j + 2, 1] = i;
        xlWorkSheet.Cells[j + 2, 2] = chart2.Series[0].Points[j].YValues[0];
    }



Excel.Range formatRange;
formatRange = xlWorkSheet.UsedRange.Columns["A"];
formatRange.NumberFormat = "dd-MM-yyyy";
//formatRange.NumberFormat = "mm/dd/yyyy hh:mm:ss";



xlWorkBook.SaveAs("D:\\Filtered.xlsx");
```

```
            xlWorkBook.Close(true, misValue, misValue);
            xlApp.Quit();

            releaseObject(xlWorkSheet);
            releaseObject(xlWorkBook);
            releaseObject(xlApp);


            MessageBox.Show("Excel file created , you can find the file D:\\Filtered.xlsx");
        }

        private void releaseObject(object obj)
        {
            try
            {
                System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);
                obj = null;
            }
            catch (Exception ex)
            {
                obj = null;
                MessageBox.Show("Exception Occured while releasing object " +
ex.ToString());
            }
            finally
            {
                GC.Collect();
            }
        }
    }
```

```
}
```

Form1.Designer.cs

```
using System.Windows.Forms;
using Excel = Microsoft.Office.Interop.Excel;

namespace WindowsFormsApp1
{
    partial class Form1
    {
        /// <summary>
        /// Обязательная переменная конструктора.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        /// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }
```

```
        #region Код, автоматически созданный конструктором форм Windows

        /// <summary>
        /// Требуемый метод для поддержки конструктора — не изменяйте
        /// содержимое этого метода с помощью редактора кода.
        /// </summary>
        private void InitializeComponent()
        {
            System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea1 = new
System.Windows.Forms.DataVisualization.Charting.ChartArea();
            System.Windows.Forms.DataVisualization.Charting.Legend legend1 = new
System.Windows.Forms.DataVisualization.Charting.Legend();
            System.Windows.Forms.DataVisualization.Charting.Series series1 = new
System.Windows.Forms.DataVisualization.Charting.Series();
            System.Windows.Forms.DataVisualization.Charting.ChartArea chartArea2 = new
System.Windows.Forms.DataVisualization.Charting.ChartArea();
            System.Windows.Forms.DataVisualization.Charting.Legend legend2 = new
System.Windows.Forms.DataVisualization.Charting.Legend();
            System.Windows.Forms.DataVisualization.Charting.Series series2 = new
System.Windows.Forms.DataVisualization.Charting.Series();
            this.chart1 = new System.Windows.Forms.DataVisualization.Charting.Chart();
            this.button1 = new System.Windows.Forms.Button();
            this.comboBox1 = new System.Windows.Forms.ComboBox();
            this.chart2 = new System.Windows.Forms.DataVisualization.Charting.Chart();
            this.comboBox2 = new System.Windows.Forms.ComboBox();
            this.comboBox3 = new System.Windows.Forms.ComboBox();
            this.checkedListBox1 = new System.Windows.Forms.CheckedListBox();
            this.button2 = new System.Windows.Forms.Button();
            this.comboBox4 = new System.Windows.Forms.ComboBox();
            this.button3 = new System.Windows.Forms.Button();
```

```
        this.button4 = new System.Windows.Forms.Button();
        ((System.ComponentModel.ISupportInitialize)(this.chart1)).BeginInit();
        ((System.ComponentModel.ISupportInitialize)(this.chart2)).BeginInit();
        this.SuspendLayout();
        //
        // chart1
        //
        chartArea1.Name = "ChartArea1";
        this.chart1.ChartAreas.Add(chartArea1);
        legend1.Name = "Legend1";
        this.chart1.Legends.Add(legend1);
        this.chart1.Location = new System.Drawing.Point(128, 12);
        this.chart1.Name = "chart1";
        series1.ChartArea = "ChartArea1";
        series1.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
        series1.Legend = "Legend1";
        series1.Name = "До фільтрації";
        this.chart1.Series.Add(series1);
        this.chart1.Size = new System.Drawing.Size(564, 397);
        this.chart1.TabIndex = 0;
        this.chart1.Text = "chart1";
        //
        // button1
        //
        this.button1.Location = new System.Drawing.Point(12, 22);
        this.button1.Name = "button1";
        this.button1.Size = new System.Drawing.Size(110, 23);
        this.button1.TabIndex = 1;
        this.button1.Text = "Додати з EXCEL";
```

```
        this.button1.UseVisualStyleBackColor = true;
        this.button1.Click += new System.EventHandler(this.Button_import);
        //
        // comboBox1
        //
        this.comboBox1.Items.AddRange(new object[] {
        "1",
        "2",
        "3",
        "4",
        "5",
        "6",
        "7"});
        this.comboBox1.Location = new System.Drawing.Point(12, 51);
        this.comboBox1.Name = "comboBox1";
        this.comboBox1.Size = new System.Drawing.Size(110, 21);
        this.comboBox1.TabIndex = 2;
        this.comboBox1.Text = "Масштабування";
        this.comboBox1.SelectedIndexChanged += new
System.EventHandler(this.comboBox1_SelectedIndexChanged);
        //
        // chart2
        //
        chartArea2.Name = "ChartArea1";
        this.chart2.ChartAreas.Add(chartArea2);
        legend2.Name = "Legend1";
        this.chart2.Legends.Add(legend2);
        this.chart2.Location = new System.Drawing.Point(670, 12);
        this.chart2.Name = "chart2";
        series2.ChartArea = "ChartArea1";
```

```
        series2.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
        series2.Legend = "Legend1";
        series2.Name = "Після фільтрації";
        this.chart2.Series.Add(series2);
        this.chart2.Size = new System.Drawing.Size(578, 397);
        this.chart2.TabIndex = 3;
        this.chart2.Text = "chart2";
        //
        // comboBox2
        //
        this.comboBox2.FormattingEnabled = true;
        this.comboBox2.Items.AddRange(new object[] {
        "Без фільтрації",
        "Фільтрування"});
        this.comboBox2.Location = new System.Drawing.Point(12, 279);
        this.comboBox2.Name = "comboBox2";
        this.comboBox2.Size = new System.Drawing.Size(121, 21);
        this.comboBox2.TabIndex = 4;
        this.comboBox2.Text = "Без фільтрації";
        this.comboBox2.SelectedIndexChanged += new
System.EventHandler(this.comboBox2_SelectedIndexChanged);
        //
        // comboBox3
        //
        this.comboBox3.Items.AddRange(new object[] {
        "1",
        "2",
        "3",
        "4",
```

```
            "5",
            "6",
            "7"});
            this.comboBox3.Location = new System.Drawing.Point(12, 306);
            this.comboBox3.Name = "comboBox3";
            this.comboBox3.Size = new System.Drawing.Size(110, 21);
            this.comboBox3.TabIndex = 5;
            this.comboBox3.Text = "Масштабування";
            this.comboBox3.SelectedIndexChanged += new
System.EventHandler(this.comboBox3_SelectedIndexChanged);
            //
            // checkedListBox1
            //
            this.checkedListBox1.CheckOnClick = true;
            this.checkedListBox1.FormattingEnabled = true;
            this.checkedListBox1.Location = new System.Drawing.Point(12, 90);
            this.checkedListBox1.Name = "checkedListBox1";
            this.checkedListBox1.Size = new System.Drawing.Size(120, 154);
            this.checkedListBox1.TabIndex = 6;
            this.checkedListBox1.SelectedIndexChanged += new
System.EventHandler(this.checkedListBox1_SelectedIndexChanged);
            //
            // button2
            //
            this.button2.Location = new System.Drawing.Point(12, 250);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(75, 23);
            this.button2.TabIndex = 7;
            this.button2.Text = "Uncheck";
            this.button2.UseVisualStyleBackColor = true;
```

```
        this.button2.Click += new System.EventHandler(this.Button_filter);
        //
        // comboBox4
        //
        this.comboBox4.FormattingEnabled = true;
        this.comboBox4.Location = new System.Drawing.Point(13, 334);
        this.comboBox4.Name = "comboBox4";
        this.comboBox4.Size = new System.Drawing.Size(121, 21);
        this.comboBox4.TabIndex = 8;
        this.comboBox4.Text = "Вибір кроку ";
        this.comboBox4.SelectedIndexChanged += new
System.EventHandler(this.comboBox4_SelectedIndexChanged);
        //
        // button3
        //
        this.button3.Location = new System.Drawing.Point(13, 362);
        this.button3.Name = "button3";
        this.button3.Size = new System.Drawing.Size(75, 23);
        this.button3.TabIndex = 9;
        this.button3.Text = "Фільтр";
        this.button3.UseVisualStyleBackColor = true;
        this.button3.Click += new System.EventHandler(this.Button_export);
        //
        // button4
        //
        this.button4.Location = new System.Drawing.Point(1137, 362);
        this.button4.Name = "button4";
        this.button4.Size = new System.Drawing.Size(75, 23);
        this.button4.TabIndex = 10;
        this.button4.Text = "Експорт";
```

```
    this.button4.UseVisualStyleBackColor = true;
    this.button4.Click += new System.EventHandler(this.button4_Click);
    //
    // Form1
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(1239, 417);
    this.Controls.Add(this.button4);
    this.Controls.Add(this.button3);
    this.Controls.Add(this.comboBox4);
    this.Controls.Add(this.button2);
    this.Controls.Add(this.checkedListBox1);
    this.Controls.Add(this.comboBox3);
    this.Controls.Add(this.comboBox2);
    this.Controls.Add(this.chart2);
    this.Controls.Add(this.comboBox1);
    this.Controls.Add(this.button1);
    this.Controls.Add(this.chart1);
    this.Name = "Form1";
    this.Text = "Form1";
    this.Load += new System.EventHandler(this.Form1_Load);
    ((System.ComponentModel.ISupportInitialize)(this.chart1)).EndInit();
    ((System.ComponentModel.ISupportInitialize)(this.chart2)).EndInit();
    this.ResumeLayout(false);

}

#endregion
```

```
private System.Windows.Forms.DataVisualization.Charting.Chart chart1;
private System.Windows.Forms.Button button1;
private System.Windows.Forms.ComboBox comboBox1;
private System.Windows.Forms.DataVisualization.Charting.Chart chart2;
private ComboBox comboBox2;
private ComboBox comboBox3;

void copychart()
{
   System.IO.MemoryStream myStream = new System.IO.MemoryStream();
   chart1.Serializer.Save(myStream);
   chart2.Serializer.Load(myStream);
   chart2.Series["До фільтрації"].Name = "Після фільтрації";
}

void checkedclear()
{
   copychart();
   int numPoints = chart2.Series[0].Points.Count;
   int list = comboBox4.SelectedIndex;
   int nu = numPoints / list;
   int nume = numPoints % list;

   for (int i = num - 1; i >= 0; i--)
   {
      for (int j = list - 1; j >= 0; j--)
      {

         if (checkedListBox1.GetItemChecked(j) == true) {
chart2.Series[0].Points.RemoveAt(j+(i*list)+nume); }
```

```
            }
        }
        if (nume > 0)
        {
            for (int j = list - 1; j >= 0 && nume > 0; j--, nume--)
            {

                if (checkedListBox1.GetItemChecked(j) == true) {
chart2.Series[0].Points.RemoveAt(nume - 1); }
            }
        }
    }


    private CheckedListBox checkedListBox1;
    private Button button2;
    private ComboBox comboBox4;
    private Button button3;
    private Button button4;
  }

}
```

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Додаток Г

ЗАТВЕРДЖУЮ
Перший проректор Українського державного
університету науки і технологій
Анатолій РАДКЕВИЧ
10.06.22

ПРОГРАМА
ГРЕБІШКОВА ФІЛЬТРАЦІЯ

Опис програми
ЛИСТ ЗАТВЕРДЖЕННЯ
44165850.01247-01 13-01-ЛЗ

Представники
підприємства-розробника
Завідувач кафедри КІТ
Вадим ГОРЯЧКІН
10.06.22

Керівник розробки
Віктор ШИНКАРЕНКО
10.06.22

Виконавець
Володимир ПОГРЕБНЯК
10.06.22

Нормоконтролер
Олена КУРОП'ЯТНИК
10.06.22

2022

ЗАТВЕРДЖЕНО
44165850.94208-01 13 01-ЛЗ

ПРОГРАМА
ГРЕБІШКОВА ФІЛЬТРАЦІЯ

Опис програми

44165850.94208-01 13 01

Листів 15

2022

АНОТАЦІЯ

Документ 44165850.94208-01 13 01 «Програма Гребішкова фільтрація. Опис Програми» входить до складу програмної документації на програму, що надає можливість користувачам фільтрувати часові ряди.

У даному документі представлений текст програм. Програми написані на мовах C#. Об’єм пам’яті, що необхідний для виконання програми, складає 1.5 ГБ. Програма функціонує в середовищі Windows.

1. ЗАГАЛЬНІ ВІДОМОСТІ

Програмний засіб «ГРЕБІШКОВА ФІЛЬТРАЦІЯ» написаний на мові C# потребує наступне програмне забезпечення:

- операційну систему MS WINDOWS 7 або вище;

2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Програмний продукт надає можливість вводити дані з таблиць програми EXEL. Та дає змогу фільтрувати дані за потрібними датами.

3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1. Алгоритм програми

При проектуванні був використаний структурний підхід.

Сутність структурного підходу до розробки ПЗ полягає в його декомпозиції (розбитті) на автоматизовані функції: система розбивається на функціональні підсистеми, які в свою чергу діляться на підфункції, що підрозділяються на задачі і так далі.

На рис. 1 приведений алгоритм програми.

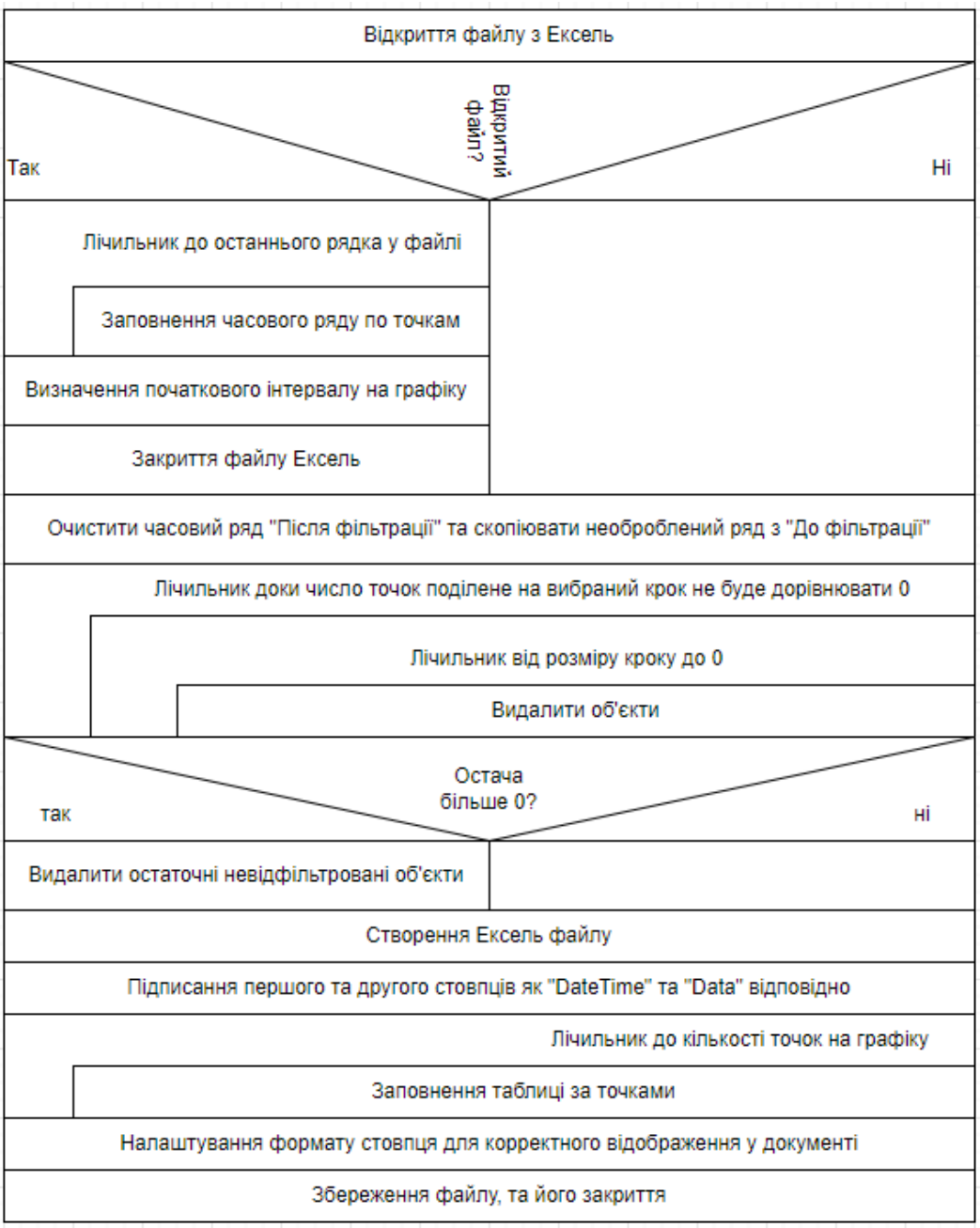

Рисунок 3.1. – Алгоритм програми

3.2. Використані методи

Структурний підхід володіє наступними перевагами:

- структурне програмування дозволяє значно скоротити кількість варіантів побудови програми з однієї і тієї ж специфікації, що значно знижує складність програми і, що важливіше, полегшує розуміння її іншими розробниками;
- у структурованих програмах, логічно пов'язані оператори знаходяться візуально ближче, а слабко пов'язані далі, що дозволяє обходитися без блок-схем та інших графічних форм зображення алгоритмів (по суті, сама програма є власною блок-схемою);
- сильно спрощується процес тестування та налагодження структурованих програм.

3.3. Структура програми

Проведення огляду вхідних і вихідних даних, формалізація задачі, розробка структур файлів і структури правил переходу.

Приведення опису структурного проектування, проектування інтерфейсу користувача, ескізи програми, вибір мови програмування.

3.4. Зв'язки програми з іншими програмами

Програма не маяє зв’язків з іншими програмами та ос.

4. ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

У роботі програма, що розробляється, розрахована на використання на IBM-сумісних персональних комп’ютерах, що мають наступні характеристики:

– Операційна система Windows
– клавіатура;
– маніпулятор миша;
– наявність CD/DVD приводу, USB роз’єму або LAN-адаптеру для встановлення необхідного ПЗ.

5. ВИКЛИК І ЗАВАНТАЖЕННЯ

Для виклику програми необхідно запустити файл «Comb filter.exe» .

6. ВХІДНІ ДАНІ

Вхідними даними програми є:

1. Дати та значення для побудови графіку, дати які потрібно відфільтрувати.
2. Дані вводяться з файлу таблиць формату EXEL.

7. ВИХІДНІ ДАНІ

Часовий ряд, та його відфільтрована версія у вигляді графіку та файл з відфільтрованими даними.

8. ОПИС ІНТЕРФЕЙСУ КОРИСТУВАЧА

Користувач може мати доступ швидкого виконання дій. Після запуску програми з’явиться головна сторінка програми з графічним зображенням, де буде знаходитись часовий ряд, та будуть кнопки:

- «Додати з EXEL» - введення даних з таблиць EXCEL
- «Фільтр» - введення потрібних даних для фільтрування ряду, запуск фільтрації
- «Експорт» - експорт EXCEL файлу.

Усі функціональні можливості ПЗ знаходиться на головній сторінці.

Для завершення роботи програми, необхідно закрити програму натиснувши «Х» згори з права.

9. ПОРЯДОК РОБОТИ З ПРОГРАМОЮ

Перед початком роботи потрібно підготувати EXCEL файл для його подальшої фільтрації. У файлі потрібно у стовпець «А» вписати потрібні дати, та напроти них у стовпці «В» записати відповідні числа. Далі можна перейти до роботи з додатком. У відчиненому вікні можна побачити інтерфейс. Спершу потрібно додати створений файл EXCEL натиснувши на кнопку «Додати з EXCEL». Для зручності можна обрати масштабування графіку. Далі, є випадаючий список, де написано «Без фільтрації», натискаємо туди та обираємо «Фільтрувати». Далі можна буде вибрати розмір кроку також у випадаючому списку. Коли вже будо обрано файл, масштаб та розмір кроку, можна обрати точки які потрібно прибрати. Після того, як користувач обрав точки, можна натискати на кнопку «Фільтр». Відбудеться фільтрація. Відфільтрований часовий ряд з’явиться на графіку справа, зліва знаходиться графік до фільтрування. Після закінчення фільтрації, користувач має змогу зберегти відфільтровані дані до таблиці EXCEL. Після натискання кнопки «Експорт» файл буде збережено до диску D.

10. ПОВІДОМЛЕННЯ

В табл. 10.1 наведені повідомлення користувачу.

Таблиця 10.1 – Повідомлення користувачу

Текст повідомлення	**Опис ситуації**	**Рекомендовані дії**
«Не вибрано файл для імпорту»	Користувач не обрав файл	Заново вибрати файл та натиснути «відкрити»