ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДТРИМКИ ЛАБОРАТОРНОГО ПРАКТИКУМУ З ДИСЦИПЛІНИ "ЕМПІРИЧНІ МЕТОДИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ" МОВОЮ C#**»

Виконав: студент 4 курсу, групи ПД–44
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Москалець О.В.
(прізвище та ініціали)

Керівник Шевченко С.М.
(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
_____Негоденко О.В.
"____" _____2023 року

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

МОСКАЛЬЦЮ ОЛЕКСАНДРУ ВАСИЛЬОВИЧУ
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка програмного забезпечення для підтримки лабораторного практикуму з дисципліни "Емпіричні методи інженерії програмного забезпечення" мовою C#»_____
_____
Керівник роботи: Шевченко С.М., к.п.н., доцент кафедри ІПЗ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26.
2. Строк подання студентом роботи «01» червня 2023 року
3. Вхідні дані до роботи
3.1 Робоча програма навчальної дисципліни «Емпіричні методи програмної інженерії, навчальні посібники та підручники
3.2 Методи аналізу даних: кореляційний аналіз, дисперсійний аналіз, кластерний аналіз
3.3 Науково-технічна література з питань, пов'язаних щодо створення Windows Form (інтерфейс програмування додатків)_____
_____
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
4.1 Огляд тем дисципліни «Емпіричні методи програмної інженерії»

4.2 Розробка програмного забезпечення для підтримки вивчення тем дисципліни «Емпіричні методи програмного забезпечення»
4.3 Об'єктно – орієнтована мова програмування C# та інтерфейс програмування додатків

5. Перелік демонстраційного матеріалу (назва основних слайдів)
5.1 Аналіз аналогів математичних пакетів
5.2 Вимоги до програмного забезпечення
5.3 Програмні засоби реалізації
5.4 Діаграма варіантів використання
5.5 Діаграма класів
5.6 Діаграма станів
5.7 Екранні форми

6. Дата видачі завдання «25» лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Обговорення напрямку дослідження та обрання теми	10.04.23	Виконано
2	Підбір науково-технічної літератури	18.04.23	Виконано
3	Вимоги до системи	25.04.23	Виконано
4	Створення лінійної кореляції	01.05.23	Виконано
5	Створення рангової кореляції	05.05.23	Виконано
6	Створення нелінійної кореляції	12.05.23	Виконано
7	Створення кластерного аналізу	15.05.23	Виконано
8	Створення дисперсійного аналізу	16.05.23	Виконано
9	Вступ, висновки, реферат	17.05.23	Виконано
10	Розробка обов'язкових демонстраційних матеріалів	22.05.23	Виконано
11	Попередній захист роботи	26.05.23	Виконано
12	Здача роботи	01.06.23	Виконано

Студент _____ Москалець О.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шевченко С.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 88 с., 33 рис., 3 табл., 29 джерел.
ЛАБОРАТОРНИЙ ПРАКТИКУМ, ЕМПІРИЧНІ МЕТОДИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ, КЛАСТЕРНИЙ АНАЛІЗ, КОРЕЛЯЦІЙНИЙ АНАЛІЗ, МОВА C#, MICROSOFT VISUAL STUDIO

Об'єкт досліджень - процес навчальної діяльності студентів при вивченні дисципліни «Емпіричні методи програмної інженерії».

Предмет дослідження – програмне забезпечення для лабораторного практикуму з дисципліни «Емпіричні методи програмної інженерії».

Мета роботи - підтримка лабораторного практикуму з дисципліни «Емпіричні методи програмної інженерії» за рахунок використання програмного забезпечення мовою C#.

Дане дослідження присвячене питанню підтримки навчальної діяльності студентів у процесі вивчення дисципліни Емпіричні методи програмної інженерії. У роботі проведено аналіз існуючих математичних пакетів, таких як Microsoft Excel, MATLAB, MathCAD, Wolfram Mathematica методів розрахунку комп'ютерної алгебри з метою визначення їх можливостей і недоліків, на основі яких сформульовані вимоги до розробки нового додатку.

Додаток був реалізований в інтегрованому середовищі розробки Microsoft Visual Studio, використовуючи інтерфейсне програмування додатків Windows Form, мовою програмування C#. Додаток працює в операційній системі Windows.

Даний додаток може бути використано у всіх сферах, де використовується кореляційний аналіз (лінійна, рангова та нелінійна кореляція) та кластерний аналіз.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення

ЗВО – заклад вищої освіти

ЕМПІ – емпіричні методи програмної інженерії

UML – Unified Modeling Language

SDLC – Software Development Life Cycle

MSDN – Microsoft Developer Network

MS – Microsoft

IDE – Integrated Driver Electronics

OCL – Object Constraint Language

OMG – Object Management Group

XMI – Metadata Interchange

ВСТУП

Емпіричні дослідження використовуються для розкриття емпіричних питань, які потребують точного формулювання на основі наявних даних. Зазвичай дослідник вже має певну теоретичну базу, на якій ґрунтується його дослідження. З цієї теорії випливають припущення або гіпотези. Ці гіпотези потім використовуються для прогнозування конкретних подій. Прогнози, що випливають з цих гіпотез, можуть бути підтверджені або спростовані за допомогою відповідних експериментів. Залежно від результатів експерименту, теорії, на яких ґрунтуються гіпотези та прогнози, можуть бути підтверджені або спростовані [1].

Актуальність теми. Розробка програмного забезпечення для підтримки лабораторного практикуму з дисципліни "Емпіричні методи інженерії програмного забезпечення" мовою C# є актуальною темою, особливо з урахуванням швидкого розвитку інформаційних технологій та важливості практичного досвіду для студентів, які вивчають цю дисципліну.

Емпіричні методи програмної інженерії є досить специфічними та вимагають практичних навичок для їх освоєння. Розробка програмного забезпечення для підтримки лабораторного практикуму дозволяє студентам отримати практичний досвід у застосуванні емпіричних методів та інструментів в реальних проектах [1].

Мова програмування C# є популярною в індустрії розробки програмного забезпечення, особливо для розробки десктопних, веб- та мобільних додатків на платформі .NET. Використання мови C# у розробці програмного забезпечення для підтримки лабораторного практикуму дозволяє студентам ознайомитись з сучасними інструментами та практиками, які вони зможуть використовувати в майбутній професійній діяльності [2].

У програмній інженерії виникає багато проблем, пов'язаних з розрахунком методів класифікації даних.

Метою дослідження є підтримка лабораторного практикуму з дисципліни «Емпіричні методи програмної інженерії» за рахунок використання програмного забезпечення мовою C#.

Предметом дослідження є програмне забезпечення для лабораторного практикуму з дисципліни «Емпіричні методи програмної інженерії».

Об'єктом досліджень є процес навчальної діяльності студентів при вивченні дисципліни «Емпіричні методи програмної інженерії».

Завданням роботи:

- Провести огляд та аналіз тем дисципліни Емпіричні методи програмної інженерії.
- Проаналізувати характеристики існуючих аналогів математичних пакетів з метою визначення їх можливостей для підтримки навчальної діяльності студентів.
- Сформулювати вимоги до програмного забезпечення з урахуванням недоліків існуючих засобів.
- Провести огляд та аналіз IT-засобів для розробки програмного забезпечення кваліфікаційної роботи бакалавра.
- Спроектувати та розробити програмне забезпечення для розрахунку кореляційного та кластерного аналізу.
- Провести тестування розробленого програмного забезпечення.

Методи дослідження:

- Для визначення теоретичних основ дослідження використовувалися такі методи: аналіз психолого-педагогічної, методичної й спеціальної літератури; огляд програм, підручників, навчальних посібників з вищої математики та інформатики для ЗВО; аналіз ресурсів Інтернет.
- Для отримання емпіричних даних були використані такі методи: спостереження, бесіди з викладачами математичних дисциплін.
- Для обробки й аналізу результатів були застосовані статистичні методи.
- методи кластерного аналізу;
- методи регресійного аналізу;
- методи кореляційного зв'язку;
- методи класифікації даних у програмній інженерії.

Практична значущість результатів отриманих досліджень є розробка додатку, який допомагає студенту краще засвоїти методи аналізу даних та їх застосування у процесі вивчення дисципліни "Емпіричні методи програмної інженерії".

Структура роботи складається із переліку умовних скорочень, вступу, трьох розділів з підрозділами, висновку, список використаних джерел та демонстраційного матеріалу (презентація).

1. РОЛЬ ЛАБОРАТОРНОГО ПРАКТИКУМУ З ЕМПІРИЧНИХ МЕТОДІВ ПРОГРАМНОЇ ІНЖЕНЕРІЇ ЯК СКЛАДОВОЇ ПРАКТИЧНОЇ ПІДГОТОВКИ СТУДЕНТІВ-ПРОГРАМІСТІВ

1.1 Основні поняття і визначення. Методика проведення емпіричних досліджень

Зазвичай, представлені емпіричні дослідження, які включають збір і аналіз даних і досвіду з метою характеристики, оцінки та виявлення зв’язків між результатами розробки програмного забезпечення, практиками та технологіями.

Емпіричні дослідження включають спостереження та дослідження конкретних явищ, проведення експериментів, а також узагальнення, класифікацію та опис результатів дослідження та експерименту з метою їх впровадження у практичну діяльність людей [1].

Існує чотири основних напрями емпіричних досліджень, які включають:

- дослідження, пов'язані з технічною складовою, такі як аналіз оточення для розробки, дослідження використовуваних програмних методів у процесі розробки та самого програмного продукту.
- дослідження процесів розробки на індивідуальному рівні розробника, включаючи всі аспекти їхньої роботи.
- дослідження можливостей інтеграції програмних продуктів, створених різними розробниками.
- дослідження взаємодії та координації роботи команди розробників, що охоплює аспекти того, чи функціонує команда як єдиний механізм, або ж просто як група окремих розробників, що взаємодіють на певних етапах процесу.

Програмна інженерія визначається як використання системного, методичного підходу до розробки, використання та підтримки програмного забезпечення, а також вивчення цих підходів, що передбачає застосування інженерних принципів

до ПЗ. Поняття "програмна інженерія" вперше з'явилося у 1968 році на конференції з програмної інженерії, що була організована НАТО. [3].

Етапи виконання розробки ПЗ зображено на рис. 1.1.

Рисунок 1.1 — Етапи виконання ПЗ

1.2 Мета, завдання та зміст дисципліни «Емпіричні методи програмної інженерії» у підготовці майбутніх інженерів-програмістів

Метою дисципліни "ЕМПІ" у підготовці майбутніх інженерів-програмістів є надання студентам необхідних знань та навичок для застосування емпіричних методів у процесі розробки програмного забезпечення [4].

Завданнями цієї дисципліни є:

- Навчити студентів збирати та аналізувати емпіричні дані в контексті програмної інженерії. Це включає оцінку якісних та кількісних даних, їх обробку та інтерпретацію.

- Розглянути різні методи досліджень, експериментального моделювання та статистичного аналізу, що застосовуються у програмній інженерії. Студентам надається можливість ознайомитися з різними методиками та виробити навички їх застосування.
- Ознайомити студентів зі зв'язком між емпіричними дослідженнями та практичними аспектами розробки програмного забезпечення. Вони вивчають, як результати досліджень можуть бути використані для покращення процесу розробки, виявлення проблем та прийняття обґрунтованих рішень [2].

Зміст дисципліни "Емпіричні методи програмної інженерії" включає в себе вивчення основних понять, принципів та практичних аспектів емпіричних досліджень у програмній інженерії. Студенти знайомляться з методами збору та аналізу даних, проведення експериментів, статистичного аналізу результатів, моделювання та узагальнення даних. Крім того, розглядаються питання етики та достовірності емпіричних досліджень у програмній інженерії [5].

1.3 Лабораторний практикум як різновид практичного заняття

Лабораторний практикум є важливою складовою навчального процесу в навчальних закладах, зазвичай університетах або коледжах, студенти і слухачі вперше здійснюють самостійну практичну діяльність у конкретній області. Лабораторні заняття, як і інші форми практичних занять, є посереднім між теоретичною роботою на лекціях і семінарах та застосуванням знань на практиці. Ці заняття успішно поєднують елементи теоретичного дослідження і практичної роботи [6].

Під час проведення лабораторних робіт студенти та слухачі краще освоюють програмний матеріал завдяки тому, що абстрактні формули та розрахунки стають конкретними. Це дозволяє з'єднати теорію з практикою, що сприяє зрозумінню

складних аспектів науки та розвитку професійних навичок учнів і слухачів як майбутніх фахівців.

Самі слова "лабораторія" та "лабораторний" вказують на поняття, пов'язані з застосуванням розумових та фізичних зусиль для дослідження раніше невідомих шляхів і засобів для вирішення наукових та прикладних завдань [6].

Термін "практикум", що використовується для опису певної системи практичних навчальних робіт, не випадково виражає той самий основний принцип, який походить від грецького слова "praktikos" і означає "дієвий". Тому цей термін відображає такі види навчальних занять, де вимагається від студентів підвищеної розумової активності.

Ні одна форма навчальної роботи не вимагає такого прояву ініціативи, спостережливості та самостійності в прийнятті рішень, як лабораторна робота. Тому всі кафедри навчальних закладів, включаючи загальнонаукові, загальноінженерні, технічні та спеціалізовані дисципліни, відводять значну кількість часу навчальних планів на лабораторні заняття - від 20% до 30%.

З метою поєднання теорії і практики в навчальних закладах все більш поширюються комплексні лабораторні роботи, які проводяться на підґрунті широкого технічного контексту і включають застосування різноманітного обладнання в умовах, наближених до реальних, в яких майбутні фахівці будуть працювати.

Лабораторні заняття є одним з видів самостійної практичної роботи студентів і слухачів, під час яких вони здійснюють експерименти для поглиблення і закріплення теоретичних знань, які були викладені на лекціях, в умовах лабораторії. Це дає змогу набути навичок наукового експериментування, аналізу отриманих результатів, ознайомитися з вимірювальною апаратурою та методами роботи з нею, а також сформувати початкові навички організації, планування та проведення наукових досліджень.

У всіх документах, що стосуються вищого навчального закладу, зазначається необхідність подальшого удосконалення та активізації лабораторного практикуму як основного засобу підвищення професійної підготовки майбутніх фахівців.

Важливо, щоб цей процес супроводжувався поліпшенням змісту, організації, модернізації лабораторного обладнання та методичного забезпечення [6].

1.4 Типові класи задач дисципліни «Емпіричні методи програмної інженерії»

У дисципліні "ЕМПІ" типові класи задач можуть включати наступні:

Експериментальні дослідження: Цей тип задач полягає у проведенні експериментів для збору даних про програмні системи, процеси розробки або інші аспекти програмної інженерії. Дослідження можуть включати порівняння різних підходів, оцінку ефективності методів або визначення взаємозв'язків між різними факторами [1].

Аналіз даних: Цей тип задач включає обробку та аналіз даних, отриманих від програмних проектів або розробки. Аналіз даних може включати виявлення шаблонів, трендів, кореляцій або інших характеристик, що допомагають зрозуміти аспекти програмного процесу або покращити розробку програмних систем.

Моделювання і симуляція: В цьому типі задач використовуються математичні моделі або симуляції для вивчення аспектів програмної інженерії. Моделі можуть бути використані для прогнозування результатів процесу розробки, аналізу впливу змін на систему або оцінки ризиків .

Опитування та анкетування: Цей тип задач включає проведення опитувань або анкетувань серед розробників програмного забезпечення, керівників проектів або інших учасників процесу розробки. Опитування можуть стосуватися думок, вподобань, використання практик або оцінки задоволеності.

Контентний аналіз: Цей тип задач включає аналіз текстового або мультимедійного контенту, що стосується програмного забезпечення. Контент може включати документацію, коментарі, блоги або інші види інформації. Аналізування контенту може допомогти виявити тенденції, потреби користувачів або проблеми, пов'язані з програмними системами.

Валідація та верифікація: Цей тип задач включає оцінку та перевірку якості програмного забезпечення. Валідація визначає, чи відповідає система потребам та вимогам користувачів, а верифікація перевіряє, чи відповідає система специфікаціям та стандартам .

Ці типові класи задач дисципліни "ЕМПІ" допомагають досліджувати та вдосконалювати процес розробки програмного забезпечення з використанням наукових методів та емпіричних даних [1].

1.4.1 Дисперсійний аналіз

Дисперсійний аналіз, відомий також як аналіз варіації (ANOVA), є статистичним методом, що дозволяє вивчати вплив різних факторів на залежну змінну. Цей метод був розроблений біологом Р. Фішером у 1925 році для оцінки експериментів у сфері рослинництва, але потім його важливість була визнана у психології, педагогіці, медицині та інших галузях [8].

Дисперсійний аналіз (ANOVA) застосовується для вивчення впливу одного або кількох якісних факторів на одну кількісну змінну, яка є залежною. Головною метою дисперсійного аналізу є перевірка значущості різниць між середніми значеннями шляхом порівняння дисперсій. Загальну дисперсію змінної розкладають на незалежні складові, які характеризують вплив окремих факторів або їх взаємодію [8].

Основною ідеєю дисперсійного аналізу є розгляд одних змінних як причин, а інших як наслідків. Незалежні змінні, що можуть бути контрольованими факторами, дозволяють досліднику змінювати їх значення та аналізувати отримані результати [1].

Основною метою дисперсійного аналізу (ANOVA) є вивчення значущості відмінностей між середніми значеннями шляхом порівняння дисперсій. Розподіл загальної дисперсії на різні джерела дозволяє порівняти дисперсію, обумовлену різницею між групами, з дисперсією, що виникає в межах однієї групи. Відмінно

від t-критерію Стьюдента, дисперсійний аналіз дозволяє порівнювати середні значення трьох або більше груп [8].

У процесі спостереження за досліджуваним об'єктом якісні фактори можуть змінюватися випадковим або заданим чином. Конкретна реалізація фактора (наприклад, температурний режим, вибране обладнання або матеріал) називається рівнем фактора або способом обробки. Дисперсійний аналіз з фіксованими рівнями факторів називають моделлю I, тоді як модель II включає випадкові фактори [9].

Основними схемами організації вихідних даних з двома і більше факторами є:

- Перехресна класифікація, характерна для моделей I, в яких кожен рівень одного чинника поєднується при плануванні експерименту з кожної градацією іншого чинника;
- Ієрархічна (гніздова) класифікація, характерна для моделі II, в якій кожному випадковому, навмання обраному значенням одного фактора відповідає своє підмножина значень другого чинника.

У випадку, коли одночасно досліджуються залежності відгуку від якісних і кількісних факторів, застосовується коваріаційний аналіз.

Залежно від кількості факторів, що визначають варіацію результативної ознаки, дисперсійний аналіз поділяють на однофакторний і двофакторний аналізи зображено на рис. 1.4.1.

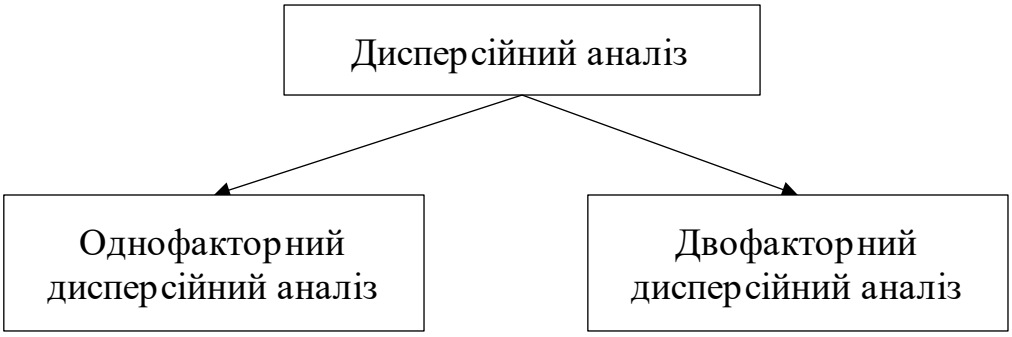


Рисунок 1.4.1 — Дисперсійний аналіз

1.4.1.1 Однофакторний дисперсійний аналіз

Основна мета однофакторного аналізу, як правило, полягає в оцінці впливу певного фактора на відповідь, яку досліджують. Додатковою ціллю може бути порівняння двох або кількох факторів для встановлення різниці в їх впливі на відповідь, що часто називається контрастом факторів. Перший крок полягає у проведенні перевірки нульової гіпотези, яка стверджує відсутність будь-якого впливу досліджуваного фактора (факторів). Ця гіпотеза означає, що зміни значень ознаки в порівнюваних вибірках є випадковими, а всі дані походять з однієї генеральної сукупності [10].

Якщо нульову гіпотезу відхиляють, наступним кроком є кількісна оцінка впливу досліджуваного фактора та будівництво довірчих інтервалів для отриманих характеристик. У випадку, коли нульову гіпотезу не можна відхилити, зазвичай її приймають і роблять висновок про відсутність впливу. Однак, якщо є підстави вважати, що такий вплив має бути присутнім (наприклад, на підставі теоретичних уявлень про об'єкт дослідження), необхідно перевірити наявність інших факторів, які можуть маскувати цей вплив.

У однофакторному дисперсійному аналізі вхідні дані можуть бути представлені у вигляді таблиць, де кількість стовпців відповідає кількості рівнів фактора, а кількість значень у кожному стовпці представляється кількості спостережень для відповідного рівня фактора (табл. 1.4.1.1). Кількість спостережень може варіювати для різних рівнів фактора. При цьому припускається, що результати спостережень для різних рівнів є вибірками з сукупностей, які мають нормальний розподіл. Такі сукупності мають однакові середні значення та дисперсії, які не залежать від рівнів фактора. Основна мета аналізу полягає в перевірці нульової гіпотези про рівність середніх значень досліджуваних сукупностей, які наведені в таблиці 1.4.1.1.

Таблиця 1.4.1.1 — Форма таблиці спостережень при проведенні однофакторного дисперсійного аналізу

Результати вимірювань	Рівні фактора			
	1	2	…	k
1	x_{11}	x_{12}	…	x_{1k}
2	x_{21}	x_{22}	…	x_{2k}
…	…	…	…	…
n_i	$x_{n_1 1}$	$x_{n_1 2}$	…	$x_{n_1 k}$

Альтернативною процедурою до однофакторного дисперсійного аналізу є ранговий однофакторний аналіз Краскела-Уолліса. Цей метод був розроблений американським математиком Вільямом Краскелом та економістом Вільсоном Уоллісом у 1952 році. Критерій Краскела-Уолліса використовується для перевірки нульової гіпотези про однаковість ефектів впливу на досліджувані вибірки з невідомими, але рівними середніми значеннями. Важливою умовою є наявність більше ніж двох вибірок. Нульова гіпотеза стверджує, що k вибірок з обсягами n_1, n_2, …, n_k отримані з однієї генеральної сукупності. Критерій Краскела-Уолліса є розширенням U-критерію Манна-Уїтні для випадку, коли кількість вибірок $k > 2$.

Рангові методи, зокрема метод Краскела-Уолліса, не вимагають нормального розподілу результатів спостережень і можуть застосовуватися як для кількісних даних з невідомим розподілом, так і для порядкових ознак [9].

У табл. 1.4.1.1 замість спостережень заносять їх ранги r_{ij}, отримані шляхом впорядкування за зростанням усієї сукупності спостережень x_{ij}. При цьому одержують табл. 1.4.1.2.

Таблиця 1.4.1.2 — Загальний вигляд вихідної таблиці рангового однофакторного аналізу

№ результату	№ вибірки			
	1	2	…	k
1	r_{11}	r_{12}	…	r_{1k}
2	r_{21}	r_{22}	…	r_{2k}
…	…	…	…	…
n_i	$r_{n_1 1}$	$r_{n_1 2}$	…	$r_{n_1 k}$

Критерій Джонкхієра - Терпстра був розроблений Т.Дж. Терпстрою і Е.Р. Джонкхієром і застосовується, коли групи результатів упорядковані за зростанням досліджуваного фактора у порядковій шкалі. Він є потужнішим за критерій Краскела - Уолліса щодо перевірки гіпотези про монотонний вплив фактора .

М-критерій Бартлетта, розроблений М.С. Бартлеттом, використовується для перевірки нульової гіпотези про однаковість дисперсій у кількох нормальних генеральних сукупностях з різними обсягами вибірок.

G-критерій Кокрена, запропонований В.Г. Кочреном, застосовується для перевірки нульової гіпотези про однаковість дисперсій k $(k>2)$ нормальних генеральних сукупностей за незалежних вибірок однакового розміру.

Непараметричний критерій Левене, запропонований Х. Левене, є альтернативою критерію Бартлетта і застосовується в ситуаціях, коли немає достатньої впевненості, що вибірки підпорядковуються нормальному розподілу.

Критерій Брауна - Форсайта, розроблений М.Б. Брауном і А.Б. Форсайтом, є більш стійким тестом, який використовує значення $Z_{ij}=|x_{ij}-x_j|$, де x_j - медіана спостережень у j-й вибірці.

Лінійним контрастом у моделі адитивного впливу фактора на відгук мається на увазі лінійна функція середніх значень k незалежних нормальних вибірок з різними, але невідомими дисперсіями.

Метод множинних порівнянь (Шеффе), запропонований Г. Шеффе, використовується для виявлення відмінностей між вибірками, які належать до певної множини даних і мають різні дисперсії [1].

1.4.1.2 Двофакторний дисперсійний аналіз

Для аналізу взаємодії двох факторів із пов'язаними нормально розподіленими вибірками застосовують двофакторний дисперсійний аналіз. Вхідні дані представляються у вигляді таблиці 1.4.1.2, де стовпці відображають дані для певного рівня перший фактор відображається у стовпцях, а другий фактор - у

рядках. Розмірність таблиці даних складає $n \times k$, де n - кількість рівнів першого фактора, а k - кількість рівнів другого фактора [10].

Таблиця 1.4.1.2 — Таблиця даних двофакторного дисперсійного аналізу

Рівні фактора B	Рівні фактора A			
1	x_{11}	x_{12}	…	x_{1k}
2	x_{21}	x_{22}	…	x_{2k}
…	…	…	…	…
n	x_{n1}	x_{n2}	…	x_{nk}

Головною відмінністю від таблиці однофакторного дисперсійного аналізу є можливість виникнення неоднорідності даних у стовпцях, особливо якщо другий фактор має значний вплив. У практичних дослідженнях часто застосовують більш складні таблиці для двофакторного дисперсійного аналізу, включаючи такі, де кожна комірка містить набір даних (повторні вимірювання), що відповідають фіксованим значенням обох факторів.

Для опису даних, представлених у таблиці 1.4.1.2, часто можна використовувати адитивну модель. Ця модель передбачає, що значення відгуку є сумою внесків кожного з факторів, позначених як b_i і t_j, а також незалежної від факторів випадкової компоненти ε_{ij}.

Якщо випадкова компонента ε_{ij} має нормальний розподіл з нульовим середнім значенням і однаковою дисперсією σ_2 для всіх значень i та j, тоді можна використовувати двофакторний дисперсійний аналіз, також відомий як дисперсійний аналіз за двома ознаками.

Якщо припущення, необхідні для застосування двофакторного дисперсійного аналізу, не виконуються, можна використовувати непараметричний ранговий критерій Фрідмана. Мілтон Фрідман, американський економіст, розробив цей критерій наприкінці 1930 року, спільно з Кендаллом та Смітом. Особливістю цього критерію є те, що він не залежить від типу розподілу даних. Єдиним передбаченням є однаковість і неперервність розподілу значень εij, а також незалежність цих значень одне від одного.

Критерій Пейджа, також відомий як L-критерій, був запропонований американським статистиком Є.Б. Пейджем у 1963 році. Він призначений для перевірки нульової гіпотези H_{01}:$\tau_1 = \tau_2 = ... = \tau_k = 0$ або H_{02}: $\beta_1 = \beta_2 = ... = \beta_n = 0$ проти альтернатив $\tau_1 \leq \tau_2 \leq ... \leq \tau_k$ або, відповідно, $\beta_1 \leq \beta_2 \leq ... \leq \beta n$, де принаймні одна з нерівностей є строгою. Для впорядкованих альтернатив цей критерій є потужнішим ніж критерій Фрідмана.

Q-критерій Кокрена, який був запропонований В. Кочреном у 1937 році, використовується в ситуаціях, якщо групи однорідних суб'єктів піддаються впливам, кількість яких перевищує два, і для яких можуть бути два варіанти відгуків - умовно-негативний (0) та умовно-позитивний (1), нульова гіпотеза стверджує, що ефекти впливу є однаковими.

Двофакторний дисперсійний аналіз дозволяє виявити наявність ефектів обробки, але не надає можливості точно визначити, на яких саме стовпцях цей ефект спостерігається [10].

1.4.2 Кореляційний та регресійний аналіз

Кореляція, яка походить від латинського слова "correlation", вказує на взаємозв'язок або співвідношення між різними речами або поняттями. У випадку кореляційного зв'язку в суспільно-економічних явищах, величина результативної ознаки залежить від багатьох факторів, які можуть впливати в різних напрямах одночасно або послідовно. Важливо зауважити, що між факторною і результативною ознаками не існує повного відповідності, а лише певне співвідношення. Кожному значенню факторної ознаки відповідає ціла група значень результативної ознаки. Для виявлення кореляційного зв'язку потрібно провести загальне порівняння факторів [11].

Кореляційний аналіз, також відомий як кореляційний метод, є методом дослідження взаємозв'язку між ознаками у генеральній сукупності, які є випадковими величинами з нормальним розподілом. Для успішного застосування кореляційного аналізу потрібні достатня кількість спостережень, наявність

факторних і результативних показників, а також їх кількісний вимір та представлення в інформаційних джерелах [11].

Кореляційний аналіз часто пов'язаний з регресійним аналізом, і тому його часто називають кореляційно-регресійним аналізом. Основними цілями кореляційного аналізу є встановлення характеру залежності між змінними, вимірювання ступеня цієї залежності та виявлення впливу факторів на результативну ознаку.

Процес кореляційного аналізу включає такі етапи:

- Виявлення причинно-наслідкових зв'язків між досліджуваними ознаками полягає у встановленні факторів та виборі найбільш впливових серед них, які мають вплив на результативний показник.
- Створення моделі кореляції та регресії включає в себе інформаційне забезпечення аналізу, вибір типу та форми зв'язку, а також побудову моделі.
- Аналіз кореляційних характеристик полягає в визначенні показників, що відображають рівень зв'язку між ознаками.

Проводиться статистична оцінка параметрів зв'язку між ознаками, здійснюється економічна інтерпретація цих результатів та оцінка значимості коефіцієнтів кореляції, що вказує, наскільки добре вибрані фактори пояснюють змінність результативного показника. Отримані результати використовуються для вирішення практичних завдань, таких як прийняття рішень, прогнозування, планування та нормування.

Таким чином, на етапі початкового аналізу встановлюються взаємозв'язки між результативними та факторними ознаками. Ці взаємозв'язки можуть відрізнятися залежно від характеру зв'язку, напряму впливу та аналітичного вираження (див. рис. 1.4.2) [11].

Рисунок 1.4.2 — Схема кореляційно-регресійного аналізу

На другому етапі проводиться оцінка початкової інформації для дослідження, використовуючи різні статистичні критерії, такі як середньоквадратичне відхилення, коефіцієнт варіації та інші. Після цього формується модель стохастичного зв'язку.

Для проведення кореляційного аналізу, ми використовуємо той самий приклад, що вже використовувався для демонстрації регресійного аналізу. По-перше, ми оцінюємо типовість та однорідність спостережень, шляхом визначення їх розподілу навколо середнього рівня. Для цього ми використовуємо такі показники, як середньоквадратичне відхилення (а) та коефіцієнт варіації (V) [11].

Рисунок 1.4.2 — Види зв'язків суспільних явищ

При формуванні кореляційної моделі потрібно визначити, чи буде це проста (парна) кореляція, де взаємозв'язок існує лише між однією результативною ознакою і одним фактором, або множинна кореляція, де взаємозв'язок спостерігається між результативною ознакою і кількома факторами. Крім того, залежно від характеру взаємозв'язку, кореляційні моделі можуть мати лінійну структуру (пряма лінія, зворотна лінійна залежність) або нелінійну структуру (криволінійна залежність) [11].

Для розв'язання рівняння множинної регресії і визначення показників множинної кореляції широко використовуються спеціальні комп'ютерні програми.

Отже, кореляційний аналіз відіграє важливу роль у економічному аналізі та вивченні суспільних явищ і процесів.

Використання кореляційного аналізу сприяє вирішенню наступних завдань:

- Встановлення характеру та міри зв'язку між досліджуваними явищами.
- Кількісна оцінка впливу окремих факторів та їх комбінації на рівень досліджуваного явища.
- Прогнозування змін показників аналізованого явища та об'єктивна оцінка господарської діяльності підприємства, шляхом розрахунку кількісних змін.

Кореляційний аналіз має велике значення у дослідженні зв'язків на виробництві, наприклад, між рівнем продуктивності праці та оснащеністю

виробничими засобами, між урожайністю і кількістю використаних добрив, між собівартістю та обсягом виробництва.

Завдяки кореляційному аналізу ми можемо більш детально вивчати взаємозв'язки економічних явищ і процесів, виявляти вплив факторів на результати господарської діяльності, а також виявляти та розраховувати потенційні резерви для підвищення ефективності виробництва. Усе це має позитивний вплив на управлінську, маркетингову та інші види діяльності, а також на прийняття економічно обґрунтованих господарських рішень [11].

1.4.3 Кластерний аналіз

Кластерний аналіз був вперше запропонований у 1939 році вченим К. Тріоном. Термін "кластер" в англійській мові означає гроно, згусток, пучок або група, і відображає суть цього методу.

У 60-х роках минулого століття кластерний аналіз зазнав значного розвитку, що стало можливим завдяки швидкісним комп'ютерам та визнанню класифікації як фундаментального методу наукових досліджень.

Кластерний аналіз - це метод статистичного дослідження, який включає збір даних про вибіркові об'єкти та їх упорядкування в однорідні, схожі групи.

Отже, основна ідея кластерного аналізу полягає в тому, що об'єкти дослідження групуються в "кластери" або групи схожих об'єктів за допомогою обчислювальних процедур. Цей підхід відрізняється від інших методів тим, що дозволяє класифікувати об'єкти одночасно за декількома ознаками. Для досягнення цієї мети використовуються показники, які враховують подібність за всіма параметрами класифікації [12].

Мета проведення кластерного аналізу полягає у виявленні існуючих структур, які проявляються через групування схожих об'єктів у кластери. Крім того, основною метою кластерного аналізу є надання структури досліджуваним об'єктам. Це означає, що методи кластеризації є необхідними для виявлення структури в даних, яку важко визначити візуально або за допомогою експертів.

Основні завдання кластерного аналізу включають:

- Розробка типології або системи класифікації досліджуваних об'єктів.
- Дослідження та визначення концептуальних моделей групування об'єктів.
- Формулювання гіпотез на підставі аналізу даних.
- Перевірка гіпотез про типи (групи), виокремлені у даних.

Для проведення кластерного аналізу необхідно виконати такі послідовні етапи:

- Відбір об'єктів для подальшої кластеризації.
- Визначення характеристик, за якими будуть оцінюватися об'єкти.
- Оцінка схожості між об'єктами.
- Застосування методів кластерного аналізу для формування груп схожих об'єктів.
- Перевірка надійності отриманих результатів кластерного аналізу.

Кожен з цих етапів відіграє суттєву роль у практичному використанні методу кластерного аналізу.

Визначення набору ознак, що використовуються для оцінки об'єктів у процесі кластерного аналізу є ключовим завданням, яке має велике значення у дослідженні.

Основна мета даного етапу полягає у виборі набору змінних, які найкраще відображають поняття подібності. При виборі цих ознак необхідно враховувати теоретичні положення класифікації та мету дослідження [12].

У кластерному аналізі для визначення подібності об'єктів використовуються чотири види коефіцієнтів: коефіцієнти кореляції, віддалі, асоціативності та ймовірності. Кожен з цих показників має свої переваги та недоліки, які потрібно врахувати. Найбільш розповсюдженими в соціальних та економічних науках є коефіцієнти кореляції та віддалі.

Після проведення аналізу вхідних даних формуються однорідні групи, в яких об'єкти всередині групи подібні між собою за певним критерієм, а об'єкти з різних груп відрізняються один від одного.

Кластеризацію можна здійснювати двома основними способами: ієрархічними або ітераційними процедурами.

Ієрархічні процедури передбачають послідовне формування кластерів різного рангу, які підпорядковані один одному за певною ієрархією. Зазвичай використовуються об'єднувальні дії, які включають на початковому етапі аналізу об'єднання подібних об'єктів, побудова дендрограми та формування окремих кластерів. На завершальному етапі всі об'єкти об'єднуються в одну велику групу [12].

Ітераційні процедури включають утворення одного рангу кластерів з початкових даних. Один із найпоширеніших методів є метод *k*-середніх, запропонований Дж. МакКуіном у 1967 році.

Цей метод включає такі кроки:

- розподіл даних на певну визначену кількість кластерів;
- обчислення середніх значень кожного кластера;
- розрахунок відстаней між об'єктами і центрами кластерів;
- формування нових центрів тяжіння та нових кластерів.
- Найпоширеніші методи формування кластерів включають в себе:
- метод одиничного зв'язку;
- метод повного зв'язку;
- метод середнього зв'язку;
- метод Уорда.

Метод одиничного зв'язку приєднує об'єкт до кластера, якщо він близький до хоча б одного представника цього кластера.

Метод повного зв'язку вимагає, щоб об'єкт був достатньо подібним (не менше граничного рівня) до будь-якого іншого об'єкта в кластері, щоб бути включеним.

Метод середнього зв'язку базується на середній відстані між об'єктом, який розглядається для включення в кластер, та представниками існуючого кластера.

Метод Уорда визначає приєднання об'єктів до кластерів з мінімальним зростанням внутрішньогрупової суми квадратів відхилень. Це дозволяє формувати кластери з приблизно однаковим розміром та формою [12].

Отже, науковці вважають, що кластерний аналіз має велике значення для проведення аналітичних досліджень, оскільки він дозволяє перетворити великий обсяг та різноманітний набір інформації на упорядкований і компактний вигляд. Такий підхід сприяє покращенню візуалізації, зрозуміння та сприйняття результатів аналізу, а також надає основу для здійснення прогнозів. [12].

2. ВИБІР ТЕХНОЛОГІЙ І СЕРЕДОВИЩА РОЗРОБКИ ЛАБОРАТОРНОГО ПРАКТИКУМУ З ЕМПІРИЧНИХ МЕТОДІВ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

2.1 Етапи життєвого циклу програмного забезпечення

Сучасний спосіб життя знаходить полегшення завдяки складним технологіям, таким як інтелектуальні будинки, можливості віртуальної реальності та улюблені онлайн-сервіси. Всі ці нововведення працюють на основі складного програмного забезпечення, яке функціонує непомітно в фоновому режимі.

Незалежно від складності програмного додатка, його важливою характеристикою є гнучкість і простота використання. Цього досягається завдяки ретельній підготовці кожної фази життєвого циклу розробки продукту від команди розробників.

SDLC, або життєвий цикл розробки програмного забезпечення, є послідовним набором етапів, які потрібно пройти для розробки проекту, починаючи з ініційованої ідеї й закінчуючи введенням продукту на ринок та його подальшою підтримкою [13].

Переваги, які надає система життєвого циклу розробки (SDLC), включають:

- Чіткі цілі

Проект має чітко сформульовані цілі, які розробники повинні досягти визначеними термінами. Це сприяє зменшенню ризику витрати часу та ресурсів.

- Перевірка перед впровадженням

На етапі перевірки гарантується, що програмне забезпечення успішно пройшло необхідні тести перед його введенням на ринок.

- Стрічка кроків

Розробники не можуть переходити до наступного етапу, доки попередній етап не буде завершений і затверджений менеджером проекту та замовником.

- Адаптивність членів команди проекту

Система життєвого циклу розробки програмного забезпечення (SDLC) використовує зрозумілу технічну і керівну документацію, що дозволяє команді проекту бути гнучкою. Навіть в разі втрати одного ключового члена команди, це не загрожує затримкам у проекті [13].

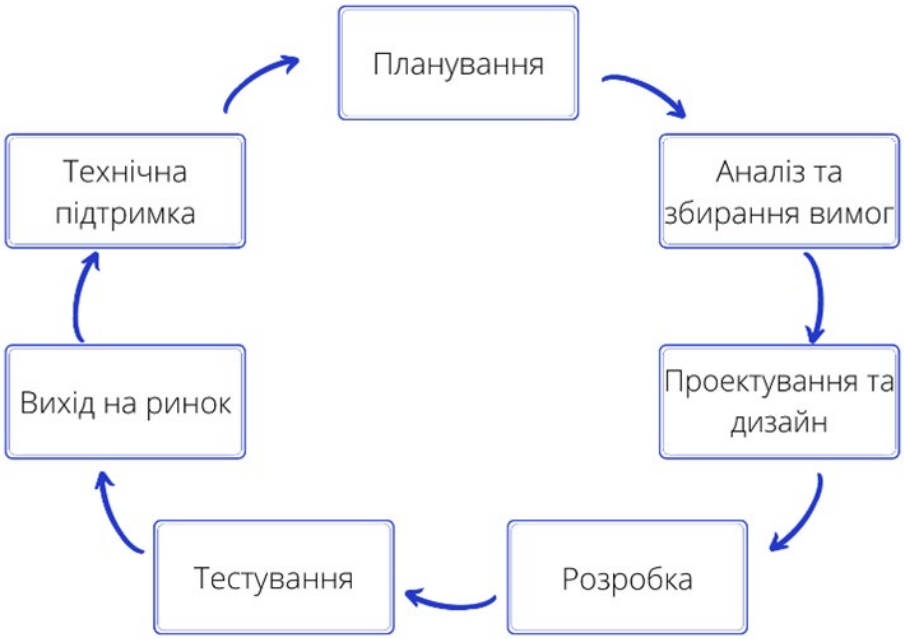


Рисунок 2.1 — Етапи розробки програмного забезпечення

Життєвий цикл розробки веб-рішень складається з семи основних фаз:

- Фаза планування
- Фаза аналізу та визначення вимог
- Фаза проектування і прототипування
- Фаза розробки програмного забезпечення
- Фаза тестування програмного забезпечення
- Фаза введення продукту на ринок
- Фаза експлуатації та технічного обслуговування

1. Планування

   Метою цього етапу є уявити, яким результатом має стати продукт.

На даному етапі, команда експертів та зацікавлених сторін визначає спільні деталі щодо майбутнього продукту, вирішуючи ключові питання. Вони зосереджуються на розумінні цілей майбутнього продукту, вирішенні проблем, які він повинен вирішити, і його призначенні. Після відповідей на ці питання всі учасники формують спільне і правильне уявлення про майбутній продукт. Це полегшує процес розробки та допомагає мінімізувати ризики, пов'язані зі створенням програмного забезпечення.

2. Аналіз та збирання вимог

Метою цього етапу є зібрати і описати всі вимоги до майбутнього програмного забезпечення.

Етап аналізу та збору вимог є одним з найбільш відповідальних етапів у життєвому циклі програмного забезпечення. На цьому етапі команда експертів співпрацює з командою замовників та, іноді, потенційними користувачами майбутнього продукту, щоб зібрати всю необхідну інформацію про проект. Цей процес розпочинається з маркетингових досліджень ринку і завершується визначенням технологічного стеку та функціональності продукту. Результатом цього етапу є проектна документація, яка містить усю зібрану інформацію і використовується для складання графіка та визначення термінів послідовних завдань та етапів розробки продукту.

3. Етап проектування та прототипування

Мета етапу: втілити вимоги до розробки в дизайн.

Після встановлення функціональних вимог та вибору технологічного стеку, можна приступити до проектування та дизайну. На даному етапі розробників формується майбутня архітектура проекту, використовуючи обрані технології. Вони розробляють адаптивний та зручний для користувачів дизайн, вирішують питання взаємодії між фронтендом та сервером, розробляють модулі та враховують систему безпеки ресурсу.

4. Етап розробки програмного забезпечення

Мета етапу: створення реального програмного забезпечення.

На етапі розробки фактично реалізується програмний код. Технічні спеціалісти використовують проектну документацію, прототипи, дизайн та архітектуру, щоб створити майбутню програму або веб-сайт.

Завдання розподіляються між членами команди відповідно до їхньої спеціалізації. Розробники відповідають за створення інтерфейсу та його взаємодію з сервером. Адміністратори баз даних додають необхідну інформацію до бази даних. Розробники front-end створюють чутливий інтерфейс майбутньої веб-додатку. Результатом цього етапу є робочий програмний продукт.

5. Етап тестування програмного забезпечення

Мета етапу: забезпечити відповідність готового продукту узгодженим вимогам.

Після завершення процесу розробки, наступає фаза тестування, яка є необхідною частиною життєвого циклу продукту. На цьому етапі команда перевіряє, чи відповідає розроблений додаток задуманим під час аналізу і збору вимог. Основні запитання, на які вони шукають відповіді, включають:

- Чи працюють всі функції та інтеграції належним чином?
- Чи коректно функціонують всі кнопки?
- Чи був досягнутий кінцевий результат, який було визначено у вимогах?

Якщо під час тестування виявляються проблеми або дефекти в роботі додатку, вони передаються програмістам для подальшої виправлення. Після того, як продукт пройшов відповідні корекції та налаштування, він готовий перейти до наступного етапу.

6. Використання готового продукту на ринку

Мета етапу: полягає в тому, щоб забезпечити готове програмне забезпечення кінцевим користувачам.

Після успішного завершення тестування, перевірена версія програми або веб-сайту випускається на основний сервер і представляється на ринку ("продакшн"). На цьому етапі служба підтримки або замовник збирають відгуки від користувачів. У разі виявлення помилок в продукті, які не були помічені на етапі тестування, але

відповідають вимогам до продукту, ці помилки передаються розробникам для виправлення на наступному етапі життєвого циклу.

7. Етап підтримки та технічного обслуговування.

Мета етапу: забезпечити підтримку готового продукту.

На цьому етапі проект переходить у режим технічного обслуговування, що передбачає постійний контроль за функціонуванням продукту, продуктивністю системи, системною безпекою та зношенням.

Технічні експерти вирішують проблеми, з якими можуть зіткнутися користувачі веб-ресурсу. Це включає виправлення залишкових помилок, які не були виправлені перед випуском, а також вирішення нових проблем, що виникають на основі повідомлень користувачів. У випадку великих проектів, етапи обслуговування можуть бути тривалішими порівняно з меншими проектами [13].

2.2 Методології життєвого циклу розробки

Методології управління життєвим циклом розробки програмного забезпечення сприяють систематичному та ефективному плануванню та поетапному виконанню процесу управляти процесом розробки, виконувати процес розробки, забезпечуючи його максимальну передбачуваність та зрозумілість. У кожної методології є своя унікальна приступ до процесу розробки продукту. Вибір моделі не впливає на постійність етапів життєвого циклу. Далі ми розглянемо дві найпоширеніші методології [13].

Гнучка модель (Agile)

Завдяки своєму гнучкому підходу, Agile дозволяє розробникам легко адаптуватися до змін на ринку, оскільки вона надає можливість вносити зміни до продукту на будь-якому етапі розробки. Agile є найбільш підходящим варіантом для проектів, що потребують гнучкості та швидкого реагування на зміни вимогам.

Цей метод передбачає створення продуктів через короткі цикли, на кожному з яких отримується робочий продукт з обмеженим набором функцій. У кожному

етапі розробки програмного забезпечення включені процеси проектування, розробки, тестування та розгортання на основному сервері, які призначені для презентації замовнику або фокус-групі.

Перевагою цього підходу є можливість власників продуктів спостерігати за результатами кожного короткого циклу, вносити свої коментарі та, за необхідності, вносити зміни. Таким чином, гнучкий життєвий цикл розробки програмного забезпечення відомий як неперервний процес [13].

Основні характеристики гнучкої моделі (Agile):

- Обмежені можливості, що дозволяють додавати нові функції під час розробки;
- Тестування здійснюється на всіх етапах розробки програмного забезпечення;
- Постійна взаємодія між клієнтами, розробниками ПЗ та проектом сприяє вдосконаленню версій після кожного спринту;
- Забезпечення якості є важливим процесом, на якому накладено особливий акцент.

Метод водоспаду (Waterfall)

Водоспадна модель найкраще підходить для проектів з чіткими очікуваннями та вимогами до майбутніх продуктів. На відміну від Agile, який наголошує на гнучкості, модель водоспаду означає, що порядок етапів у циклі розробки є жорстким. Згідно з цією моделлю, фахівці не можуть перейти до наступного етапу, поки не завершать попередній. Тому під час розробки не можна вносити жодних змін, і в кінці циклу існує лише одна версія готового продукту. При гнучкому підході для кожного нового циклу створюється робоча версія продукту.

Основні характеристики методу водоспаду (Waterfall):

- Строга послідовність кроків розробки;
- Перехід до наступного етапу можливий лише після завершення попереднього етапу;
- Клієнти не взаємодіють у процесі розробки;

- Будь-які зміни вносяться лише після випуску готового програмного забезпечення.

Створення ПЗ — це складна й масштабна робота, яка потребує детального планування, незалежно від обраної вами моделі розробки. Розробка веб-проекту починається зі збору вимог і послідовно проходить усі етапи життєвого циклу розробки [13].

Чітко визначений життєвий цикл розробки програмного забезпечення дозволяє командам експертів і клієнтам зрозуміти передбачуваність процесу розробки програмного продукту. Для розробників програмного забезпечення це означає розуміння того, що вони роблять і які наступні кроки; це додасть вам впевненості.

Тому кожен проект повинен мати чітко визначений життєвий цикл розробки програмного забезпечення. Оскільки це єдиний спосіб гарантувати, що отримане програмне забезпечення відповідає потребам як вашого бізнесу, так і ваших кінцевих користувачів [13].

2.3 Основні підходи до розробки лабораторного практикуму

Основні підходи до розробки лабораторного практикуму включають:

- Традиційний підхід: Цей підхід передбачає створення плану лабораторного практикуму, в якому чітко визначаються мета, завдання, послідовність етапів та необхідні матеріали. В рамках цього підходу, студентам надаються детальні інструкції, що допомагають їм виконувати лабораторні роботи. Цей підхід зазвичай використовується в традиційних університетських курсах і практикумах.
- Проблемно-орієнтований підхід: Цей підхід передбачає, що студентам надаються реальні проблеми або завдання, які вони повинні розв'язати або дослідити у рамках лабораторного практикуму. Цей підхід стимулює самостійність, творчість та розвиток аналітичних навичок у студентів. Вони

повинні виробляти власний план дослідження, збирати та аналізувати дані, та робити висновки. Проблемно-орієнтований підхід активно використовується в практикумах з природничих наук, інженерії та деяких гуманітарних дисциплін.

- Командний підхід: Цей підхід спонукає студентів працювати в команді, розподіляти завдання та співпрацювати для досягнення спільної мети. У рамках лабораторного практикуму, студенти формують команди і разом виконують завдання, використовуючи свої навички, знання та ресурси. Цей підхід допомагає розвивати навички співпраці, комунікації та лідерства.
- Відкритий підхід: Цей підхід дає студентам велику свободу у виборі теми або проекту для лабораторного практикуму. Вони можуть самостійно визначати свої цілі, завдання та методи дослідження. Цей підхід сприяє самостійності, творчості та особистому розвитку студентів [14].

Ці підходи можуть використовуватись окремо або комбінуватись в залежності від цілей, специфіки дисципліни та потреб студентів.

2.4 Опис середовища розробки Microsoft Visual Studio

Microsoft Visual Studio — це набір продуктів від Microsoft, який включає інтегроване середовище розробки програмного забезпечення (IDE) і різні інструменти. MS Visual Studio .NET є набором інструментів для розробки різних типів програм (консольних, Windows, мобільних, веб-застосувань) та сервісів. Це мультипрограмне середовище, яке підтримує кілька мов програмування, зокрема C++ та C# [15].

MSDN (Microsoft Developer Network) - це програмний продукт, який містить всі довідкові матеріали про розробки компанії Microsoft.

У MS Visual Studio кожне окреме програмне рішення представляє собою "solution" і складається з одного або кількох проектів. Можна відкрити лише одне

рішення одночасно, однак для роботи з кількома рішеннями потрібно запустити кілька вікон Visual Studio [15, 26].

Visual Studio надає шаблони для найпоширеніших типів проектів. Проекти та їх шаблони дозволяють користувачам зосередитися на реалізації певної функціональності, тоді як проекти обробляють загальні завдання керування та створення [15].

Майстер застосування використовується для створення нових проектів. Цей майстер створює проект на основі шаблону та надає користувачеві інтерфейс для створення шаблону для вихідного текстового файлу. Майстер налаштовує структуру програми, головне меню, панель інструментів і включає деякі файли заголовків.

Характеристики графічного інтерфейсу MS Visual Studio, який використовується для програмування під Windows, залежать від конфігурації системи. Розмір і форма вікон можуть бути налаштовані користувачем, щоб забезпечити оптимальне використання доступного простору для елементів, необхідних в даний момент [26].

У головному вікні Visual Studio можна виділити кілька основних компонентів:

- Меню та набір інструментальних панелей - тут розміщені команди для роботи з інтегрованою середою розробки (IDE).
- Вікно Провідника рішень (Solution Explorer) - дозволяє переглядати структуру проектів, що входять до рішення, в ієрархічному вигляді. Також можна побачити зв'язки між проектами та їх компонентами.
- Вікно редактора - використовується для написання програмного коду та підтримує функцію автодоповнення та підсвічування синтаксису.
- Вікно виведення стану (Output) - тут відображається інформація про процес збірки програми та виявлені помилки.

Розмір і розташування цих зон на екрані можна налаштувати відповідно до вподобань користувача.

Після створення проекту, на диску створюється проект вказаного типу, а також генерується рішення з вказаною назвою, яке містить посилання на проект [15].

2.4.1 Мова програмування C#

C# — є об'єктно-орієнтованою мовою програмування, її назва вимовляється як "сі шарп". Термін "шарп" походить від ноти до-дієз у музиці, що позначає підвищення на півтону. Отже, C# можна розглядати як оновлення та поліпшення мови C.

Мова C була епохальною у своєму часі і з'явилася в 70-х роках. Вона замінила асемблер і стала основою для інших мов програмування. За C пішов C++ (вимовляється як "сі-плюс-плюс"), мова, яка підтримує парадигми об'єктно-орієнтованого програмування та програмування загального призначення. C++ ще й досі є популярною мовою програмування [16].

У 1995 році компанія Sun Microsystems створила мову Java. Вона була розроблена з використанням граматики мови C++. Основною особливістю Java, яка відрізняє її від інших мов, стало наступне: програми Java спочатку перетворюються на байт-код, а потім перетворюються на машинний код. Це дозволяє мові Java бути незалежною від конкретного апаратного забезпечення. Крім того, у Java додали підтримку веб-розробки, що дозволяє програмувати клієнт-серверні застосунки, отримати дані з Інтернету та взаємодіяти з ними [16].

Microsoft потребувало змагатися з Sun і розробляти веб-технології. У Microsoft було мільйони рядків коду, написаних на внутрішніх мовах, головним чином на C і C++. Тому перехід до Java був непростим завданням. Це спонукало їх створити платформу для вебу, що привело до виникнення Microsoft.NET Framework [3].

.NET використовує власну мову проміжного коду, яка виконує ту саму функцію, що й байт-код для Java-машини. Для розробки на платформі .NET можна використовувати кілька мов, але основною мовою платформи є C# [16].

C# схожий із C++, оскільки обидва походять від мови C і мають подібні основні мовні конструкції. Новизна в C# полягає в його об'єктно-орієнтованій парадигмі, яка вводить поняття класів і об'єктів. Створення класів із властивостями є синтаксично зручним у C#. Він включає фундаментальні принципи об’єктно-орієнтованого програмування, такі як успадкування, інкапсуляція та поліморфізм.

C# — це мова, яка має сувору типізацію. Він включає автоматичне збирання сміття, звільняючи програмістів від ручного керування пам’яттю. Згодом у мову було введено динамічне зв’язування, асинхронні методи та шаблони [16].

C# схожий на Java і є його головним конкурентом. Обидва вони спираються на концепції C++, одночасно спрощуючи їх. Ці дві мови мають схожу філософію, але є технічні нюанси, які їх відрізняють. Ці відмінності виникають через те, що C# було розроблено спеціально для екосистеми Microsoft, тоді як Java задумувалася як середовище з відкритим кодом. Наприклад, існують варіації в тому, як реалізовані параметризовані типи та відсутність перевірених винятків у C# [16].

Більшість веб-програм та сервісів, пов'язаних з продуктами Microsoft, розроблено з використанням мови програмування C#. Наприклад, відомий веб-сайт stackoverflow.com написаний на C# з використанням фреймворку .NET. Аналогічним чином були розроблені сайти Microsoft і Dell.

C# широко використовується для розробки десктопних додатків для операційної системи Windows. Наприклад, Microsoft Visual Studio і Paint.NET були написані на C#. Крім того, платформа .NET, зокрема Windows Forms, лежить в основі таких програм, як Skype, Microsoft Office і Photoshop [16].

Геймдев-розробники добре знайомі з рушієм Unity, використовується для створення 2D і 3D комп'ютерних ігор. Програмування в Unity дозволяє зосередитися на створенні контенту гри, не займаючись багатьма технічними деталями. Більшість ігор, таких як Rust, Hearthstone і Fall Guys, розроблені в Unity за допомогою C#. Крім того, Unity пропонує можливість взаємодії з DirectX, набором компонентів для ігрової графіки та звуку [16].

C# також використовується для розробки мобільних додатків. Існують такі платформи, як Xamarin, які можуть виконувати код C# на різних операційних

системах, включаючи мобільні платформи Android та iOS. Нижче наведено список популярних додатків, які були написані з використанням Xamarin.

У 2018 році до програмного середовища .NET була додана бібліотека ML.NET, яка надає можливість використовувати моделі машинного навчання. Вона може бути менш документованою, ніж аналогічні бібліотеки для Python, але все ж зручна та часто використовується в реальних проектах [10].

2.4.2 Платформа Windows Forms

Windows.Forms використовується у Microsoft .NET для створення додатків з графічним інтерфейсом. Цей інструмент базується на бібліотеці класів .NET Framework і пропонує більш продуману та зручну модель програмування, ніж, наприклад, програмні інтерфейси Win32 API або MFC [17].

Зазвичай, Windows.Forms складається з різноманітних керованих бібліотек, що дозволяють виконувати всі необхідні дії для віконних додатків. Ці можливості включають обмін повідомленнями з операційною системою, відстеження подій клієнтського вікна, діалогові системи, взаємодію з мережею та багато інших.

У цьому контексті термін "форма" відноситься до видимої поверхні вікна, яка містить інформацію для кінцевого користувача, а також засоби (елементи керування), що дозволяють працювати з представленими даними або взаємодіяти з користувачем.

Оскільки Windows.Forms повинна містити сотні організованих класів, щоб задовольнити потреби розробника, .NET Framework розділений на ієрархічні розділи з власними іменами. Розділ System є кореневим і містить фундаментальні типи даних. Щоб наочно продемонструвати силу Windows.Forms, ми зосередимося на створенні простих, але значущих додатків з використанням мови C# та програмування [17, 27].

Платформа .NET Framework є технологією, здатною підтримувати розробку та виконання веб-сервісів та додатків для операційної системи Windows. Під час розробки .NET Framework було поставлено такі цілі:

- Створення єдиного об'єктно-орієнтованого програмного середовища для збереження та виконання об'єктного коду, незалежно від того, чи виконується код локально, у розподіленій мережі чи віддалено.
- Надання середовища виконання коду з мінімальними ймовірними конфліктами під час розгортання та управління версіями програмного забезпечення.
- Гарантування безпечного виконання коду, включаючи код, створений невідомими або неповністю довіреними сторонніми розробниками.
- Виключення проблем з продуктивністю при виконанні скриптів або коду, що інтерпретується.
- Забезпечення єдиних принципів розробки для різних типів програм, включаючи програми Windows та веб-програми.
- Підтримка взаємодії на основі промислових стандартів, що дозволяє інтегрувати .NET Framework код з іншими кодовими базами.

.NET Framework складається з двох основних компонентів:

Загальномовне середовище виконання (CLR) і бібліотека класів .NET Framework. Основним елементом платформи є середовище CLR. CLR можна розглядати як агента, який керує кодом під час його виконання та надає основні служби, такі як керування пам’яттю, керування потоками та віддалені операції. У той же час середовище накладає суворі обмеження на тип і виконує різні перевірки, щоб забезпечити безпеку та надійність вашого коду. Основна мета середовища виконання - це, звичайно, керувати вашим кодом [17, 27].

Код, який взаємодіє з середовищем виконання, називається керованим кодом, а код, який не взаємодіє з середовищем виконання, вважається некерованим кодом. Бібліотека класів є комплексною колекцією типів, зорієнтованих на об'єктно-орієнтоване програмування, які можна використовувати повторно для розробки різних видів додатків. Програми варіюються від простих програм командного рядка до програм графічного інтерфейсу користувача (GUI), які використовують розширені функції ASP.NET, такі як веб-форми та веб-служби XML.

.NET Framework дозволяє завантажувати середовище CLR у власний процес і розміщувати некеровані компоненти, які запускають виконання керованого коду. Це створює середовище програмування, яке забезпечує можливості як керованого, так і некерованого виконання. .NET Framework не лише обмежується невеликою кількістю базових середовищ виконання, але й підтримує незалежних постачальників, які розробляють власні базові середовища виконання [17].

Наприклад, ASP.NET використовує середовище виконання для створення масштабованого середовища для керованого серверного коду. ASP.NET безпосередньо взаємодіє з середовищем виконання, щоб забезпечити роботу програм ASP.NET і веб-служб XML [17, 27].

На наведеному нижче зображенні показано взаємозв'язок між середовищем CLR і загальною системою та бібліотеками класів, включаючи спеціальні програми. На цьому зображенні показано, як керований код працює в ширшій архітектурі (рис. 2.4.2).

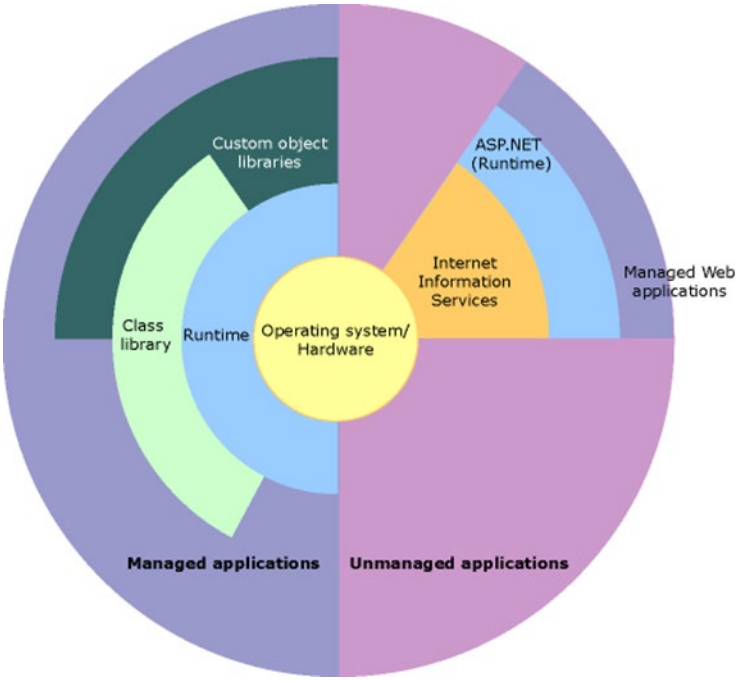


Рисунок 2.4.2 — Етапи життєвого циклу розробки програмного забезпечення

2.4.3 Клієнтські бібліотеки Accord.NET та MathNET

Accord.NET Framework — це платформа машинного навчання .NET у поєднанні з бібліотеками обробки аудіо та зображень, повністю написаними на C#. Це повна структура для побудови виробничого рівня комп'ютерного бачення, комп'ютерного прослуховування, обробки сигналів і статистичних програм навіть для комерційного використання. Повний набір зразків програм забезпечує швидкий старт, щоб швидко налагодити роботу (рис. 2.4.3) [18].

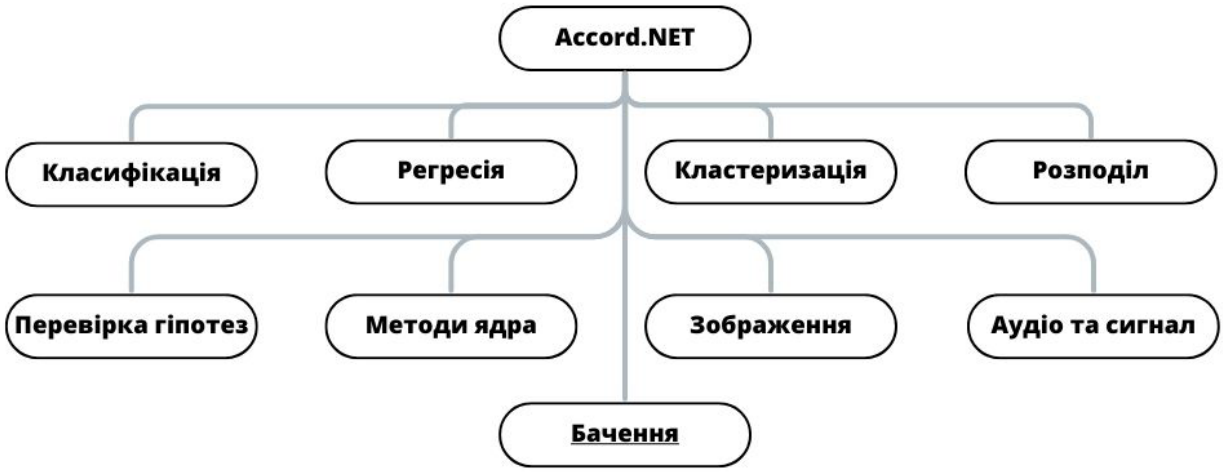


Рисунок 2.4.3 — Етапи життєвого циклу розробки програмного забезпечення

Math.NET Numerics — це бібліотека .NET і Mono Numerics з відкритим кодом, написана мовами C# і F#. Його функціональність схожа на BLAS і LAPACK.

Метою Math.NET Numerics є надання методів і алгоритмів для чисельних обчислень у науці, техніці та повсякденному житті. Розглянуті теми включають спеціальні функції, лінійну алгебру, імовірнісні моделі, випадкові числа, інтерполяцію, інтеграцію, регресію, задачі оптимізації тощо.

Math.NET Numerics є частиною ініціативи Math.NET. Доступно безкоштовно за ліцензією MIT. Націлено на Microsoft .NET 5.0, .NET 4.6.1 і вище та .NET Standard 2.0 і вище. На додаток до чисто керованих реалізацій, він також підтримує оптимізацію нативного апаратного забезпечення.

Math.NET Numerics почала об’єднувати код і команди dnAnalytics із Math.NET Iridium у 2009 році. Його натхненно ALGLIB, JAMA, Boost тощо та містить численні внески коду. Це частина ініціативи Math.NET, яка розробляє та підтримує відкриті математичні інструменти для платформи .NET з 2002 року [19].

2.5 Моделювання програмного забезпечення мовою UML

За період із середини 1970-х до кінця 1980-х років розробники почали створювати мови для об'єктно-орієнтованого моделювання, експериментуючи з різними підходами до об'єктно-орієнтованого аналізу й проектування. UML базується на кількох об'єктно-орієнтованих методах, кожен з цих інструментів спочатку був розроблений з метою підтримки окремих етапів об'єктно-орієнтованого аналізу та проектування [20].

Серед цих методів є:

- Метод, відомий як Booch, розроблений Граді Бучем, який давав гарні результати на етапах проектування та розробки програмних систем.
- Object Modeling Technique (OMT), розроблений Джеймсом Румбахом, який був ефективним для аналізу процесів обробки даних в інформаційних системах.
- Object-Oriented Software Engineering (OOSE), розроблений Айваром Джекобсоном, який надав засоби для представлення прецедентів, що мали значення на етапі аналізу вимог під час проектування бізнес-додатків [20].

У 1994-1995 роках розпочалася процес інтеграції та уніфікації цих методів, що привело до виникнення версії 0.8 уніфікованого методу (Unified Method). Розробка та підтримка мови UML були сфокусовані в рамках консорціуму OMG. Навіть якщо OMG був створений з метою стандартизації об'єктних та компонентних технологій CORBA, мова UML стала ще одним стратегічним напрямом роботи консорціуму. У листопаді 1997 року OMG оголосив UML

стандартною мовою для об'єктно-орієнтованого моделювання та взяв на себе зобов'язання його подальшого розвитку [20].

Опублікування описів нових Різні версії мови UML та процеси отримання пропозицій RFP для стандартизації є відповідальністю групи експертів. Мова UML має статус відкритого, що дозволяє приймати пропозиції щодо поліпшення та розробки. У 2003 році з'явилася публікація, що описує мову UML 2.0 в результаті обробки набору запитів RFP з 2000 року. Опис мови включав:

- інфраструктуру UML;
- OCL (Object Constraint Language) - мова обмежень об'єктів;
- суперструктура UML;
- формат для обміну даними [20].

Консорціум OMG виконує кілька ключових ініціатив під час розробки UML проекту:

- Створення моделей реального часу систем.
- Встановлення моделі виконання для детального опису поведінки моделей, які засновані на UML.
- Опрацювання даних підприємства шляхом визначення профілів, що описують розробку великих розподілених паралельних систем для підприємств.
- Установлення процесу розробки програмного забезпечення з використанням спеціалізованих структур.
- Стандартизація зберігання даних.
- Аналіз порівняння технології CORBA і мови UML.
- Використання формату XMI для обміну UML-моделями.

Основні характеристики UML діаграм:

- Умовою UML є надання користувачам готової до використання, виразної та потужної мови візуального моделювання, яка дозволяє створювати зрозумілі моделі та здійснювати обмін ними.

- У UML передбачені вбудовані механізми розширення та спеціалізації основних концепцій мови.
- Мова UML гарантує, що проекти створення програмного забезпечення є максимально незалежними від будь-якої конкретної мови програмування чи процесу розробки.
- UML надає формальну основу для чіткого тлумачення мови.
- Мова UML сприяє розширенню ринку інструментальних засобів для створення об'єктно-орієнтованого програмного забезпечення.
- UML включає найкращі методи використання мови та впровадження програмних засобів, які її підтримують [20].

2.5.1 Діаграми варіантів використання

Діаграма варіантів використання (також відома як діаграма прецедентів, сценарій використання або use case) дозволяє зобразити різні ролі та їх взаємодію з системою. Вона не відображає послідовність виконання кроків, але показує функціональні вимоги (тобто те, що система може зробити) з погляду користувача. Діаграма може бути представлена у вигляді текстового опису або графічної діаграми [21].

Діаграми використання розробляються на початковому етапі проектування системи з такими метою:

- Легке пояснення роботи системи, створення та узгодження технічного завдання (ТЗ).
- Формування функціональних вимог до системи (тобто опису того, що система повинна робити).
- Створення основи для документування та тестування системи.

Ці діаграми допомагають зрозуміти, як система буде взаємодіяти з користувачами та іншими системами, і вони є важливим інструментом для встановлення вимог до системи і розробки документації, необхідної для подальшої реалізації та тестування.

Діаграми варіантів використання складаються з чотирьох основних елементів: актор, прецедент (варіант використання), система і зв'язок.

- Актор - це поняття про особу або об'єкт, який взаємодіє з системою. Це може бути користувач, зовнішня система або інший елемент, що взаємодіє з системою.
- Прецедент (варіант використання) - це функціональна можливість або сценарій взаємодії між актором і системою. Він описує дії, які актор виконує або сценарій використання, який реалізується системою.
- Система - це об'єкт або програмне забезпечення, яке розробляється. Вона виконує функції і надає можливості для акторів.
- Зв'язок (відношення) - це зв'язок між актором і прецедентом, який показує, який прецедент виконується або взаємодіє з актором. Зв'язок може мати різні типи, такі як асоціація, включення або розширення, що деталізують взаємодію між актором і прецедентом [21].

UML діаграму варіантів використання наведено на рисунку 2.5.1.

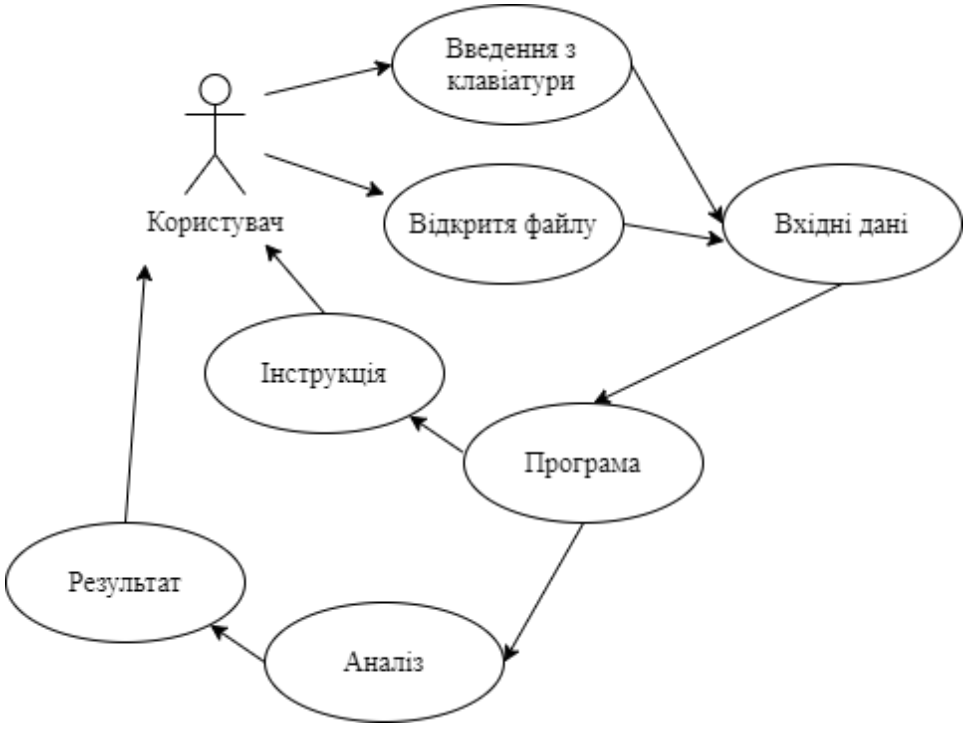


Рисунок 2.5.1 — UML діаграма варіантів використання

2.5.2 Діаграма класів

Діаграми класів є найпоширенішими діаграмами, які використовуються в UML. Вони складаються з класів, інтерфейсів, асоціацій та співпраці. Головною метою діаграм класів є візуалізація об'єктно-орієнтованої структури системи, яка має статичний характер [22].

Класи на діаграмі представляють статичний вигляд програми. Діаграми класів використовуються не тільки для візуалізації, пояснення та документування різних аспектів системи, але й для створення виконуваного коду програмних додатків.

Діаграма класів описує атрибути та операції класу та обмеження, які застосовуються до системи. Діаграми класів часто використовуються при моделюванні об'єктно-орієнтованих систем, оскільки вони можуть бути відображені безпосередньо в об'єктно-орієнтованих мовах програмування.

Зазвичай діаграми UML не мають прямого відображення в об'єктно-орієнтованих мовах програмування, але діаграма класів є винятком. Діаграма класів чітко демонструє відображення об'єктно-орієнтованих мов, таких як Java, C++, тощо. За практичним досвідом, діаграма класів зазвичай використовується для побудови програмного коду [22].

UML діаграму класів наведено на рисунку 2.5.2.

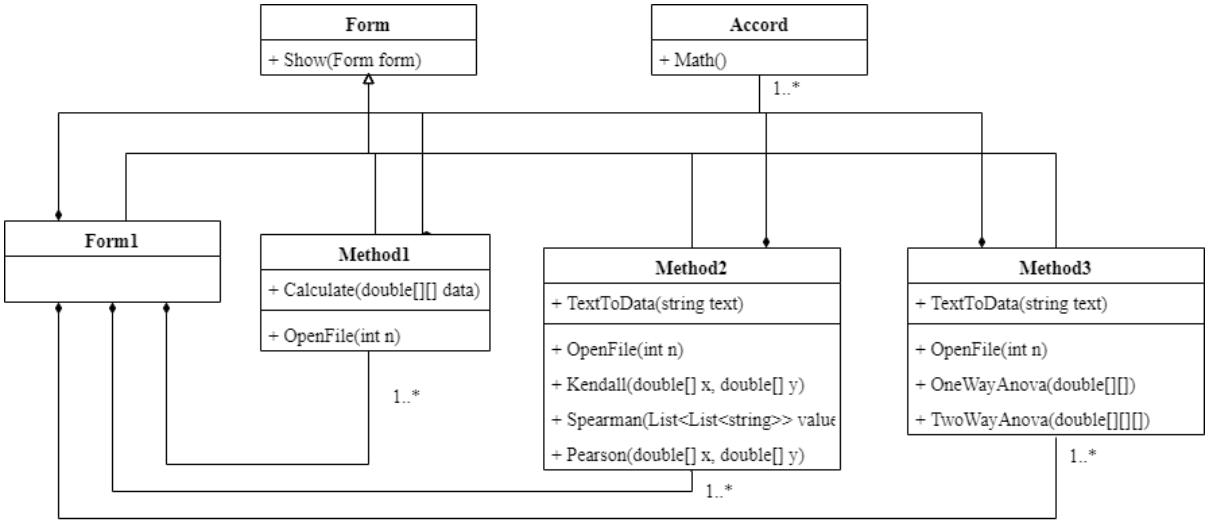


Рисунок 2.5.2 — UML діаграма класів

2.5.3 Діаграми станів

Діаграми станів є одним із п'яти видів UML-діаграм, що використовуються для моделювання динамічних аспектів системи. Вони відображають кінцевий автомат і придатні для моделювання життєвого циклу об'єкта, демонструючи потік управління від одного стану до іншого всередині конкретного об'єкта [22].

Діаграми станів можуть бути пов'язані з класами, варіантами використання або з системою в цілому, щоб візуалізувати, специфікувати, побудувати та задокументувати динаміку окремих об'єктів. Вони корисні не тільки для моделювання динамічних аспектів системи, але й для побудови виконуваних систем за допомогою прямого та зворотного проектування.

Діаграма станів (state diagram) відображає послідовність переходів між станами автомата, зосереджуючись на потоці управління від одного стану до іншого. Вона представлена у вигляді графа, де вершини відповідають станам, а дуги (ребра) показують переходи між станами.

Автомат (state machine) — це опис послідовності станів, які об’єкт проходить протягом свого життєвого циклу, як він реагує на події та як він реагує на ці події.

Стан (state) - це окрема ситуація, в якій може перебувати об'єкт під час його життєвого циклу. У кожному стані об'єкт відповідає певним умовам, виконує певні дії або очікує певні події.

Подія (event) - це специфікація значущого факту, що відбувається у певний момент часу та має певні просторові аспекти. В контексті автомата, подія викликає перехід між станами.

Перехід (transition) — це такий зв’язок між двома станами, що коли відбуваються певні події та виконуються певні умови, об’єкт у першому стані повинен виконати певні дії та перейти до другого стану.

Діяльність (activity)— це специфікація діяльності, яка відбувається всередині машини. Вона описує послідовність дій, що виконуються при перебуванні об'єкта у певному стані або під час переходу між станами.

Дія (action) - це проста обчислювальна операція, яка призводить до зміни стану моделі або повертає значення. Вона виконується під час перебування об'єкта у певному стані або під час переходу між станами [22].

Діаграма станів у мові UML представлена на рисунку 2.5.3.

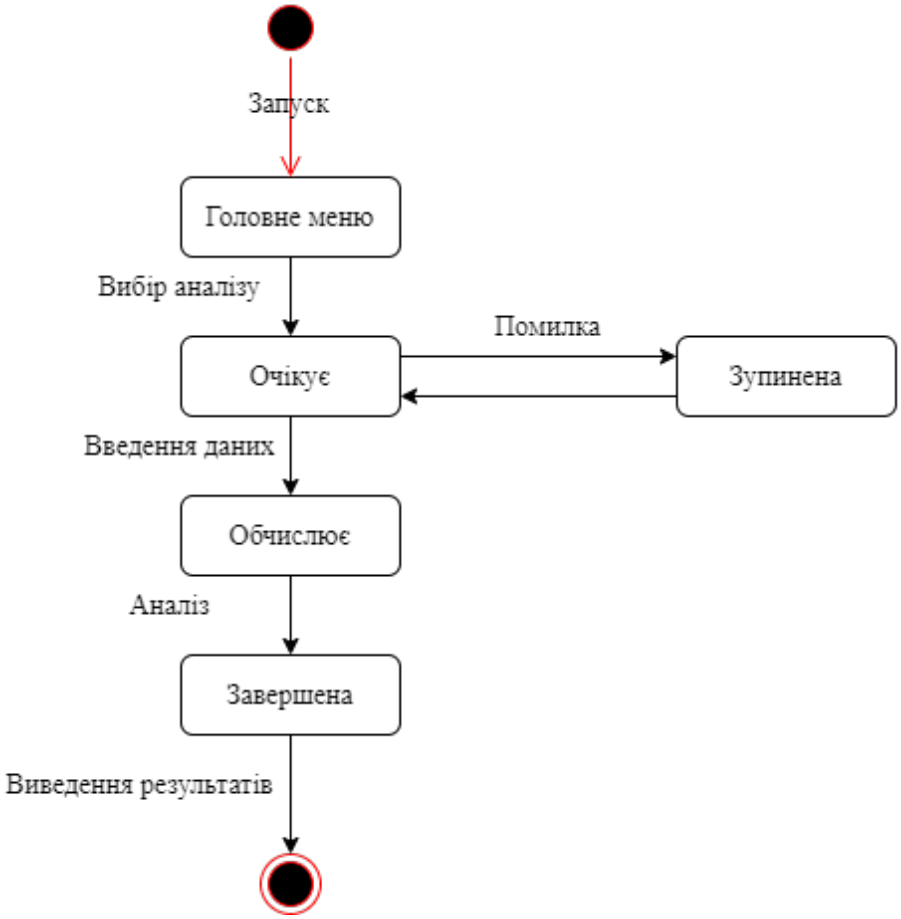


Рисунок 2.5.3 — UML діаграма станів

3. РОЗРОБКА РОЗРОБКИ ЛАБОРАТОРНОГО ПРАКТИКУМУ З ЕМПІРИЧНИХ МЕТОДІВ ПРОГРАМНОЇ ІНЖЕНЕРІЇ МОВОЮ C#

Для повного розуміння системно-структурних методів необхідно звернути увагу не лише на кінцевий результат, а й на процес прийняття рішень, який приводить до досягнення правильного результату.

Отже, було розроблено додаток, який дозволяє не лише переглядати результати введених даних, але й бачити послідовні кроки прийняття рішень, втілення яких призводить до отримання відповіді.

3.1 Інструкція роботи з програмою кластерного аналізу

При запуску програми з'являється головне меню, де можна вибрати один з трьох аналізів: кореляційний аналіз, кластерний та дисперсійний аналіз (рис. 3.1).

Рисунок 3.1 – Головне меню програми

Вибираємо кластерний аналіз розв’язування системно-структурних методів, відкривається вікно кластерного аналізу, яке зображено на рисунку 3.2.

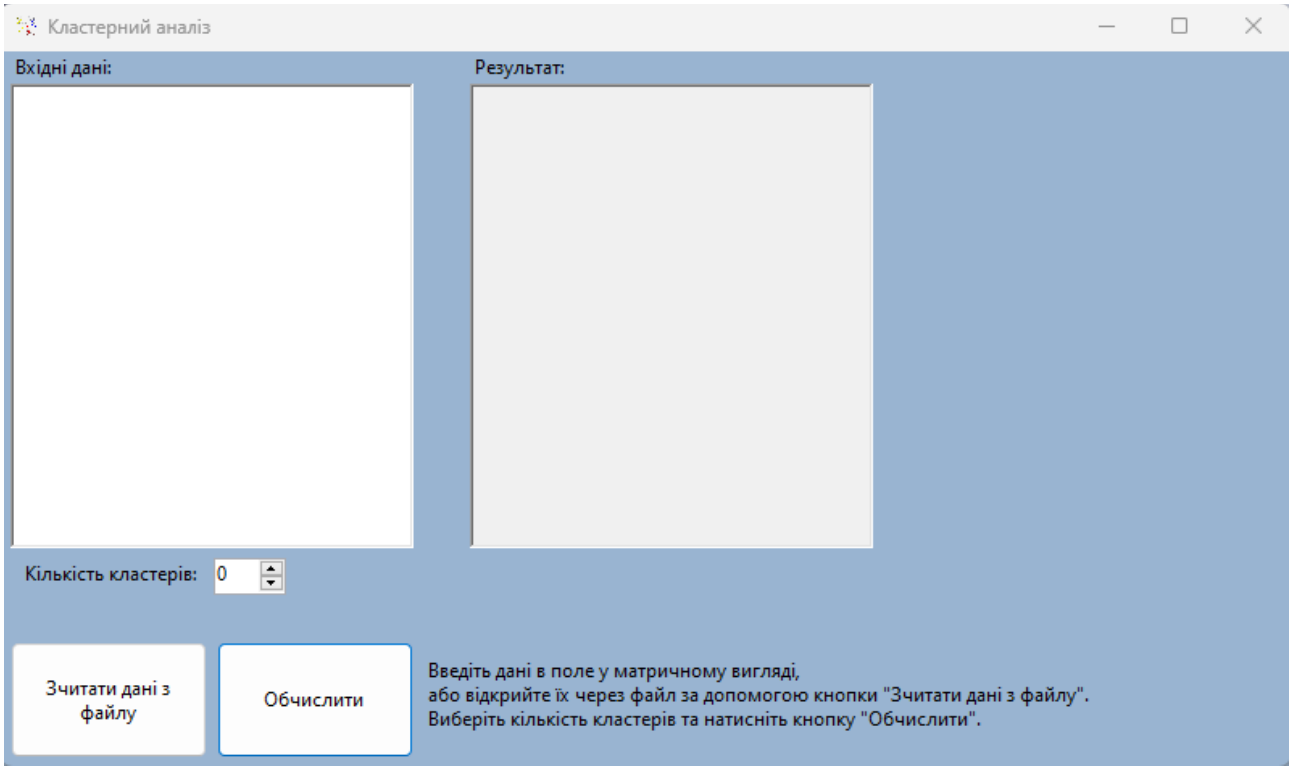


Рисунок 3.2 – Вікно кластерного аналізу програми

Для того, щоб розрахувати кластерний аналіз, треба для початку вказати потрібну кількість кластерів. В даному випадку вибираємо три кластери (рис.3.3).

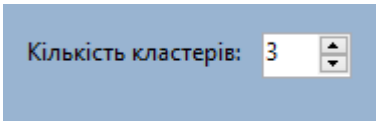


Рисунок 3.3 – Кількість кластерів

Для зразка розглянемо такі значення:

12 3 2 1 4

14 5 1 4 7

14 2 6 8 9

Після того як вказали потрібну кількість кластерів, вказуємо дані. Для задачі, яку розглядаємо, вказую відповідні значення (рис. 3.4).

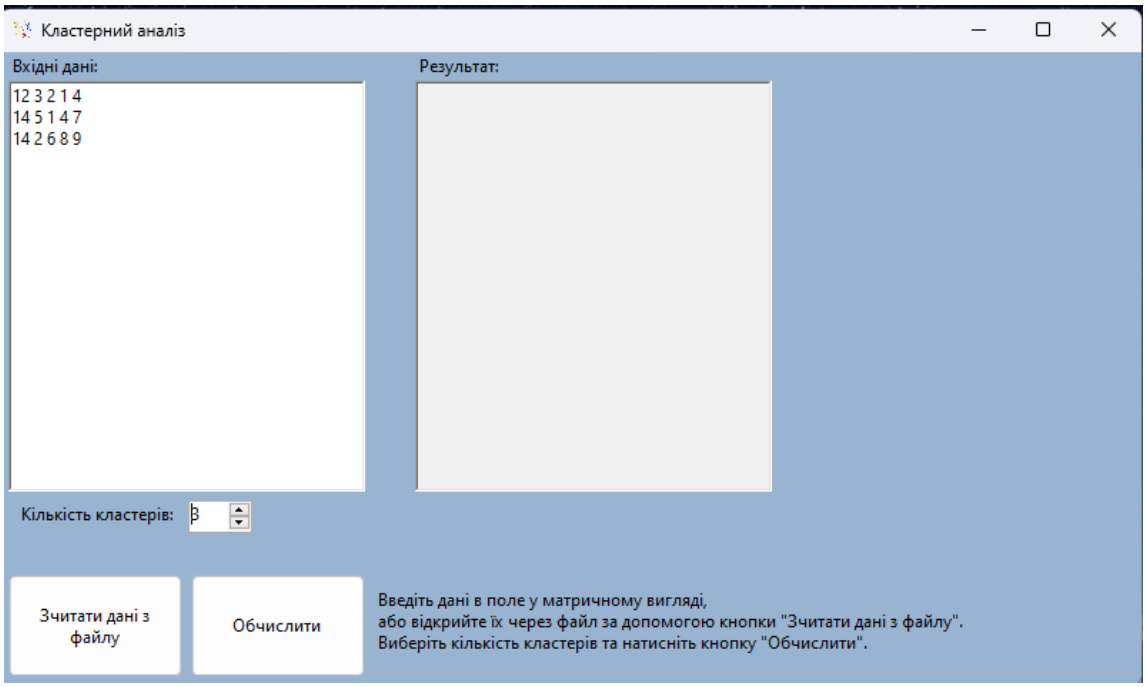


Рисунок 3.4 – Вікно кластерного аналізу програми з введеними даними

Для того щоб отримати рішення кластерного аналізу, натискаємо на поле “Обчислити” (рис.3.5).

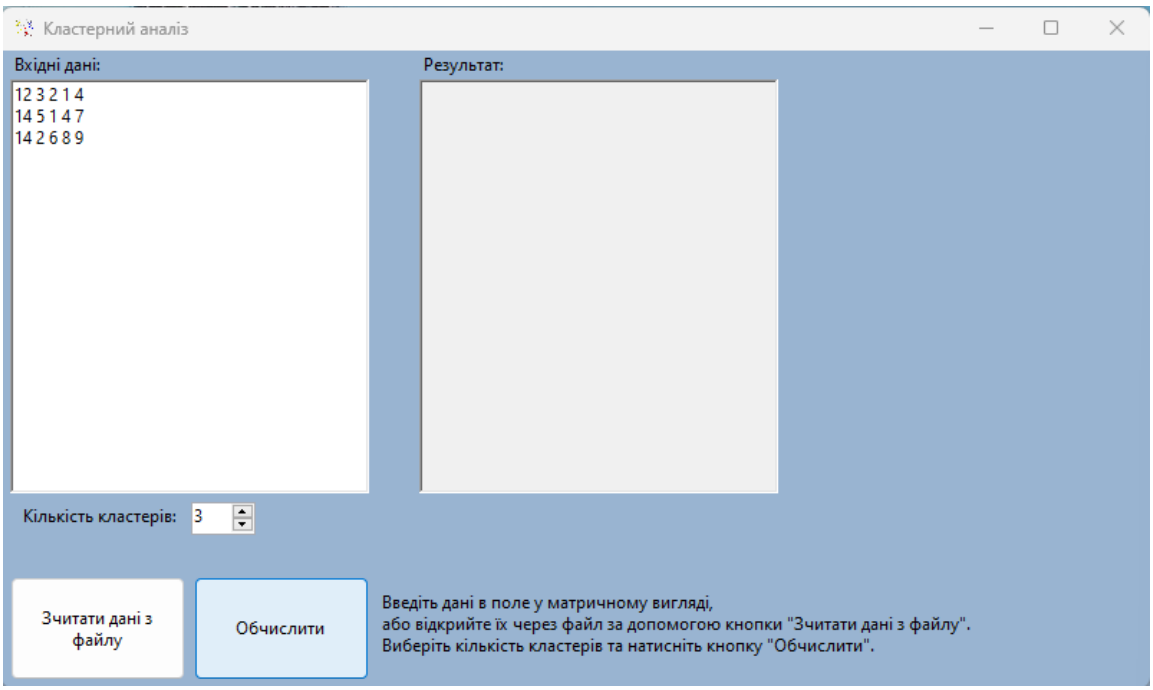


Рисунок 3.5 – Вікно натиснення на поле “Обчислити”

Після натиснення на поле “Обчислити”, рішення кластерного аналізу виводиться в поле відповіді (рис.3.6).

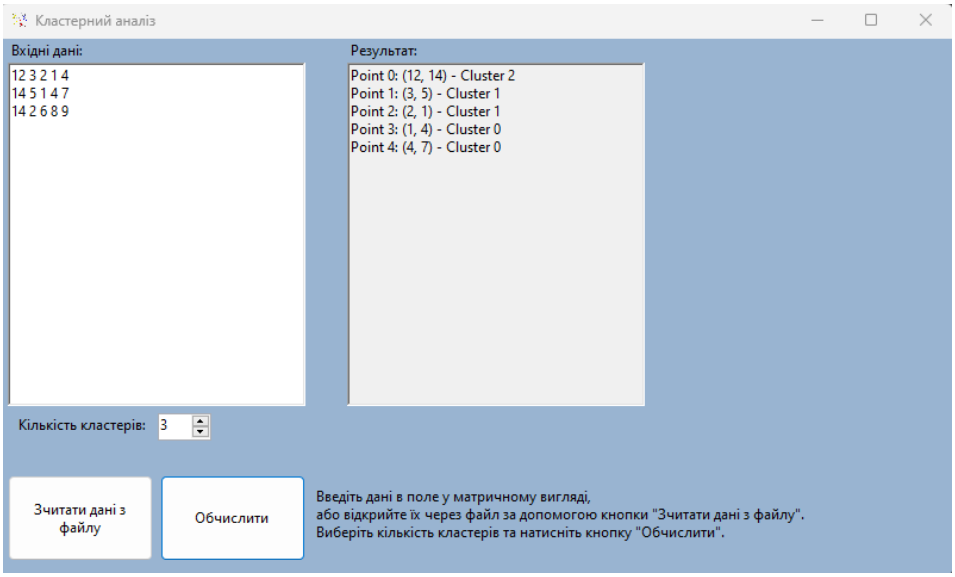


Рисунок 3.6 – Вікно рішення кластерного аналізу з рішенням прикладу

В сірому квадраті отримали розв'язок кластерного аналізу, а саме п'ять точок з координатами самих кластерів.

Також можливо створювати задачу не вводячи дані, а загружаючи їх з файлу формату «.txt» і «.csv». Для цього потрібно створити текстовий файл, який складається з данних, які треба розрахувати.

Для зразка візьмемо попередню задачу тільки вже загрузимо його з текстового файлу (рис. 3.7) та робимо аналогічні дії з розв’язуванням (Рис. 3.8).

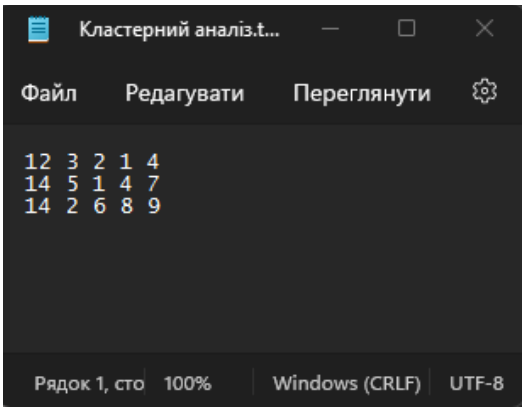


Рисунок 3.7 – Текстовий файл з прикладом

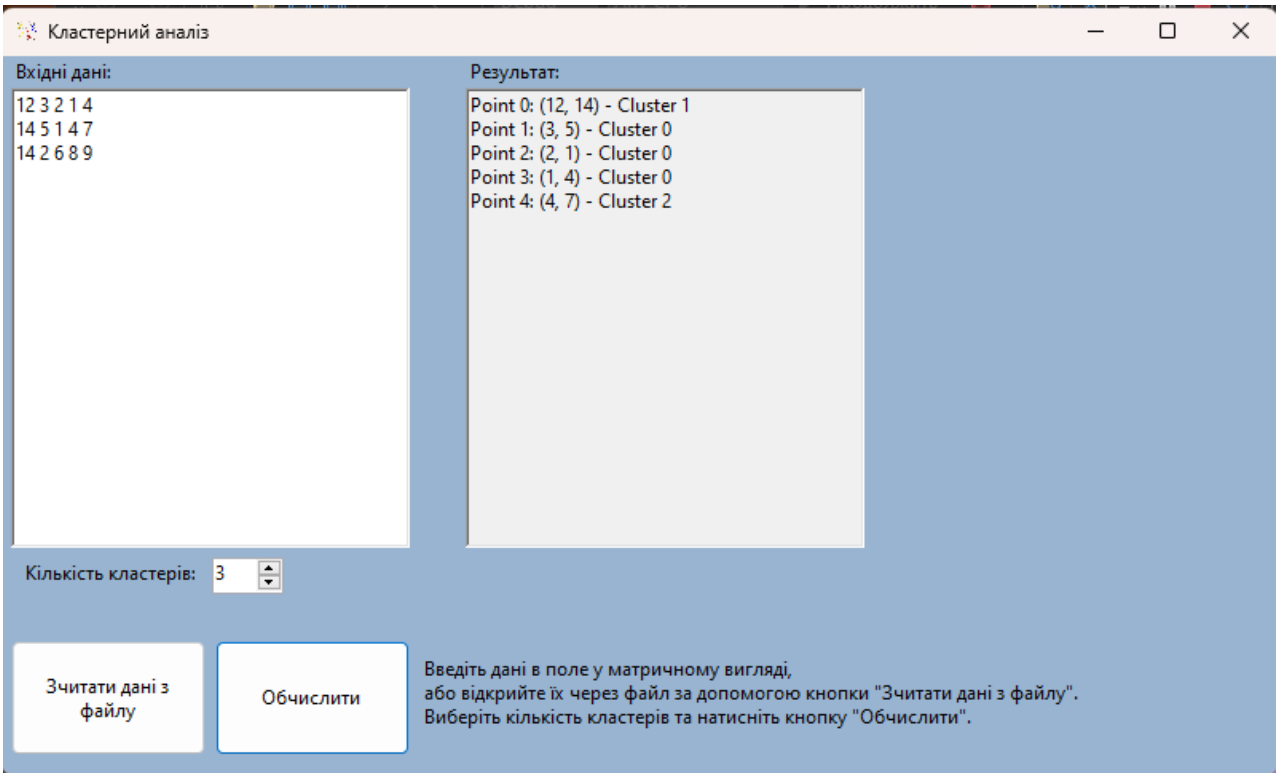


Рисунок 3.8 – Вікно рішення кластерного аналізу з рішенням прикладу

3.2 Інструкція роботи з програмою кореляційного аналізу

При запуску програми з'являється головне меню, де можна вибрати один з трьох аналізів: кореляційний аналіз, кластерний та дисперсійний аналіз (рис. 3.9).

Рисунок 3.9 – Головне меню програми

Вибираємо кореляційний аналіз розв'язування системно-структурних методів, відкривається вікно кореляційного аналізу, як бачимо є три вкладки, на яких можемо обчислити три кореляції, які зображені на рисунку 3.10.

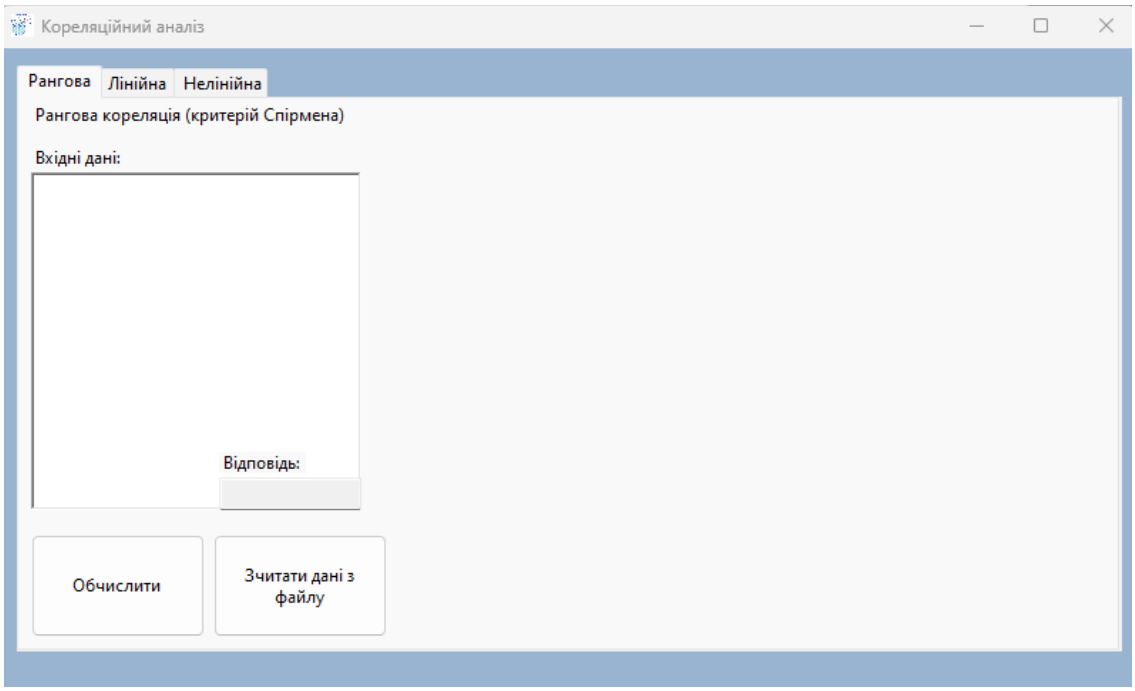


Рисунок 3.10 – Вікно кореляційного аналізу програми

Для того, щоб порахувати рангову, лінійну та нелійнійну кореляції потрібно вибрати певну вкладку та ввести дані, які треба обчислити, в нашому випадку обчислюємо рангову кореляцію (рис.3.11).

Для зразка розглянемо такі значення:

14 2 3 4 5

3 4 5 6 7

0 9 8 7 6

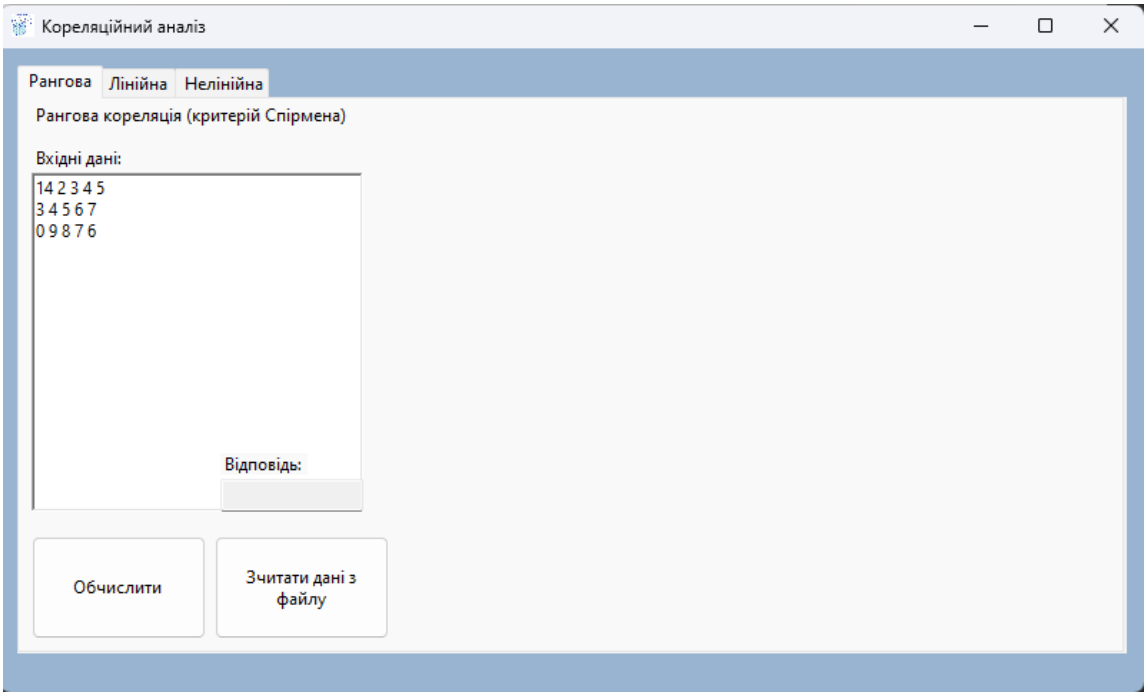


Рисунок 3.11 – Вікно вкладки рангової кореляції програми з введеними даними

Для того щоб отримати рішення рангової кореляції, натискаємо на поле “Обчислити” (рис.3.12).

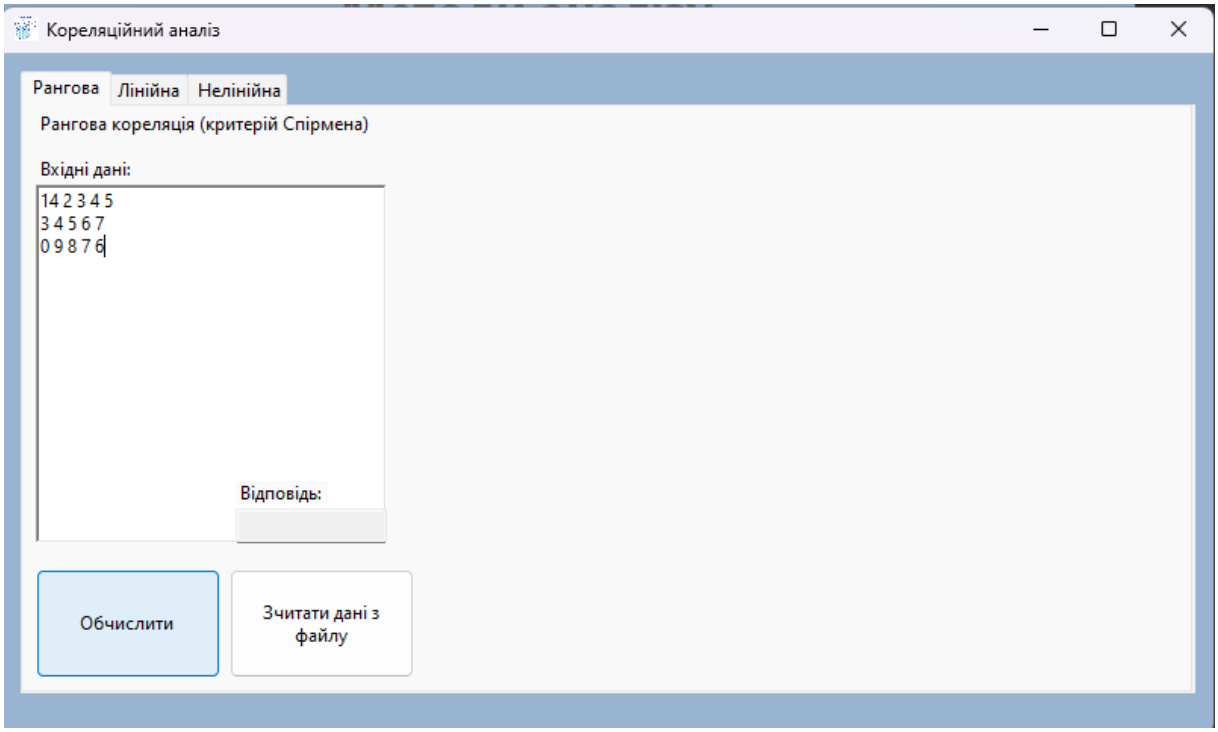


Рисунок 3.12 – Вікно натиснення на поле “Обчислити”

Після натиснення на поле “Обчислити”, рішення рангової кореляції кореляційного аналізу виведене в поле для відповіді (рис. 3.13).

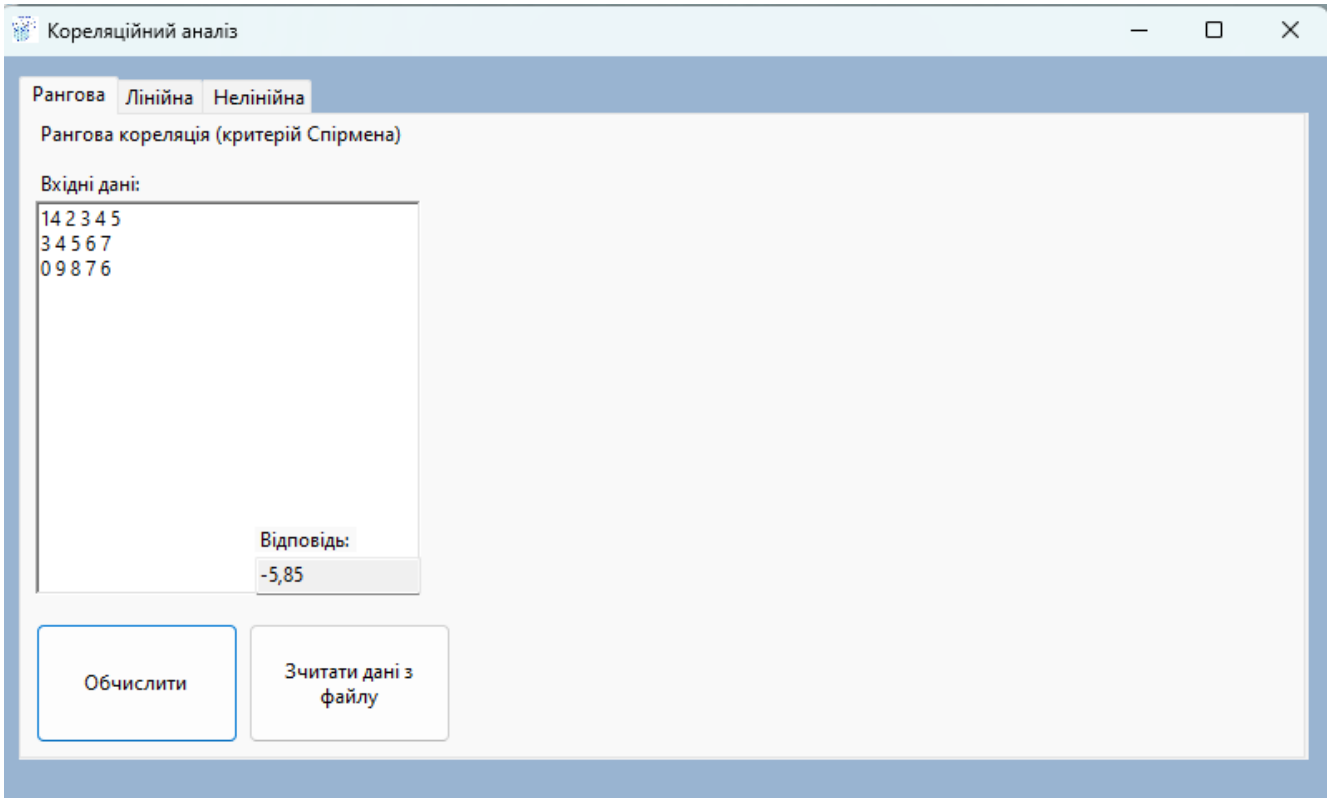


Рисунок 3.13 – Вікно рішення рангової кореляції кореляційного аналізу з рішенням прикладу

В сірому прямокутнику з підписом “Відповідь” отримали розв'язок рангової кореляції кореляційного аналізу, а саме критерій Спірмена. Для того щоб обчислити лінійну та нелійнійну кореляцію треба зробити аналогічні дії з розв’язуванням.

Також можливо створювати задачу не вводячи дані, а загружаючи їх з файлу формату «.txt» і «.csv». Для цього потрібно створити текстовий файл, який складається з данних, які треба розрахувати.

Для зразка візьмемо попередню задачу тільки вже загрузимо його з текстового файлу (рис. 3.14) та робимо аналогічні дії з розв’язуванням (Рис. 3.15).

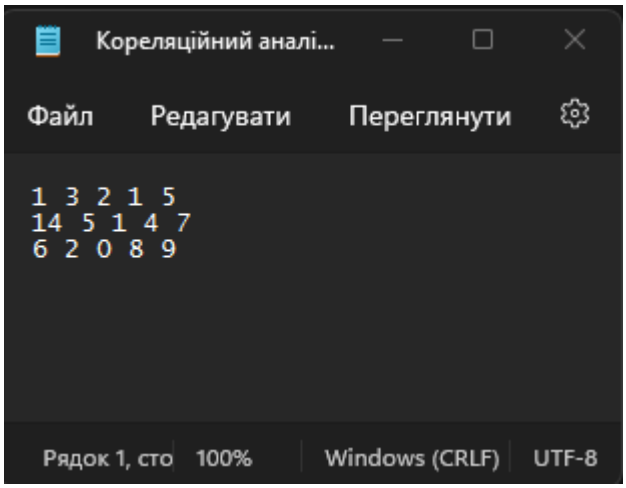


Рисунок 3.14 – Текстовий файл з прикладом

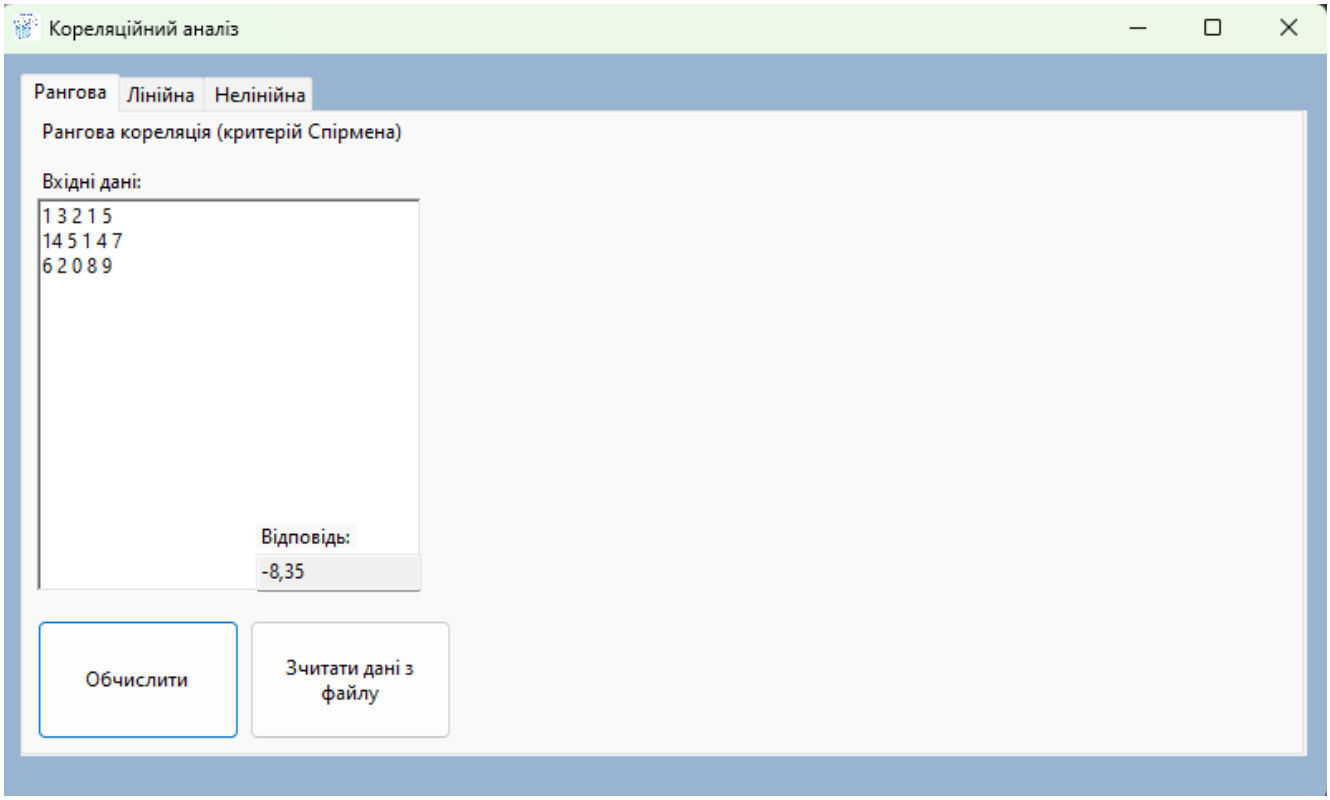


Рисунок 3.15 – Вікно рішення кореляційного аналізу з рішенням прикладу

3.3 Інструкція роботи з програмою дисперсійного аналізу

При запуску програми з'являється головне меню, де можна вибрати один з трьох аналізів: кореляційний аналіз, кластерний та дисперсійний аналіз (рис. 3.16).

Рисунок 3.16 – Головне меню програми

Вибираємо дисперсійний аналіз розв'язування системно-структурних методів, відкривається вікно кореляційного аналізу, як бачимо є дві вкладки, на яких можемо вибрати однофакторний чи двофакторний аналіз, які зображені на рисунку 3.17.

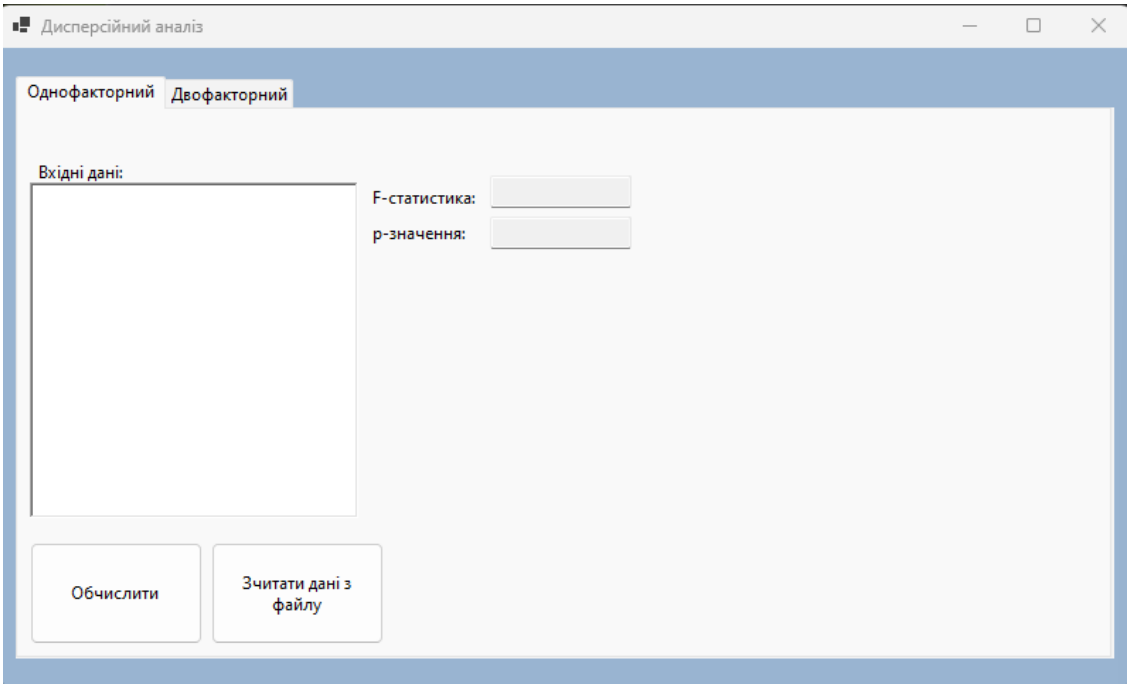


Рисунок 3.17 – Вікно дисперсійного аналізу програми

Для того, щоб порахувати однофакторний та двофакторний дисперсійний аналіз потрібно вибрати певну вкладку та ввести дані, які треба обчислити, в нашому випадку обчислюємо однофакторний дисперсійний аналіз (рис.3.18).

Для зразка розглянемо такі значення:

8 2 5 6 5

3 4 5 6 7

0 9 1 7 4

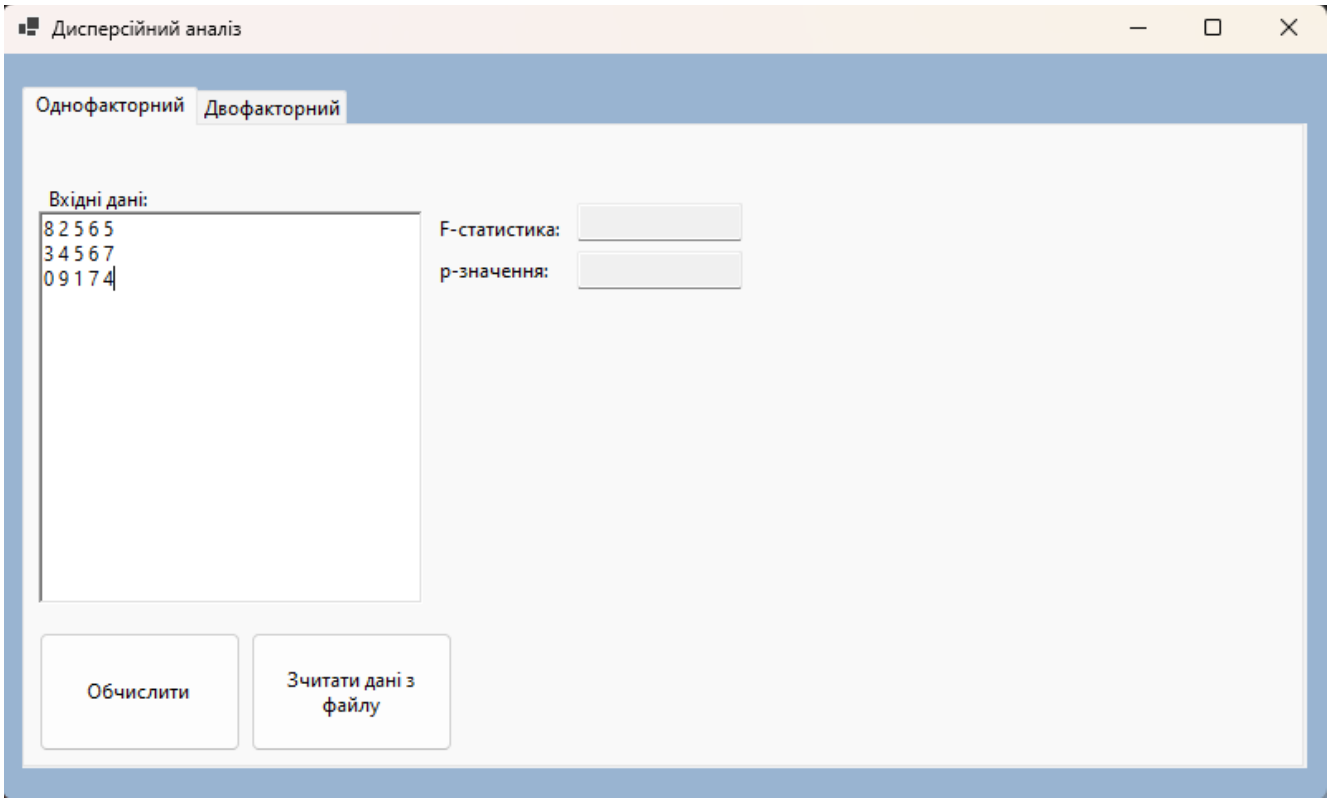


Рисунок 3.18 – Вікно вкладки однофакторного дисперсійного аналізу програми з введеними даними

Для того щоб отримати рішення однофакторного дисперсійного аналізу, натискаємо на поле “Обчислити” (рис.3.19).

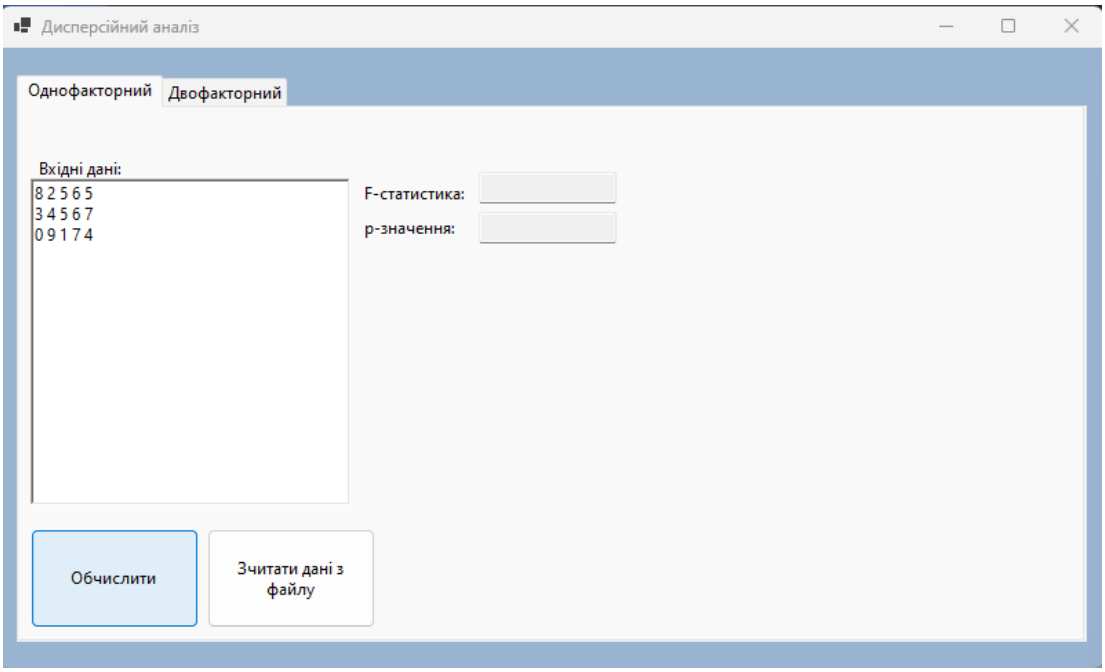


Рисунок 3.19 – Вікно натиснення на поле “Обчислити”

Після натиснення на поле “Обчислити”, однофакторного дисперсійного аналізу виведене в поле для відповіді (рис. 3.20).

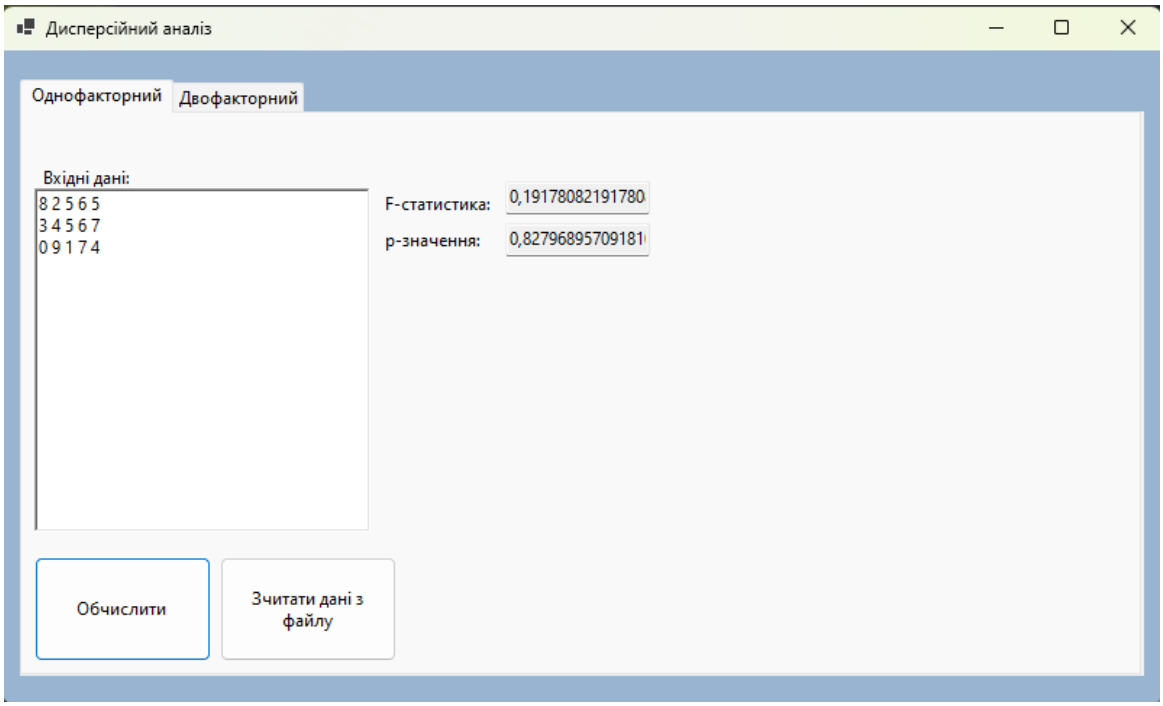


Рисунок 3.20 – Вікно рішення однофакторного дисперсійного аналізу з рішенням прикладу

В сірих прямокутниках отримали розв'язок однофакторного дисперсійного аналізу. Для того щоб обчислити двофакторний дисперсійний аналіз треба зробити аналогічні дії з розв’язуванням.

Також можливо створювати задачу не вводячи дані, а загружаючи їх з файлу формату «.txt» і «.csv». Для цього потрібно створити текстовий файл, який складається з данних, які треба розрахувати.

Для зразка візьмемо попередню задачу тільки вже загрузимо його з текстового файлу (рис. 3.21) та робимо аналогічні дії з розв’язуванням (Рис. 3.22).

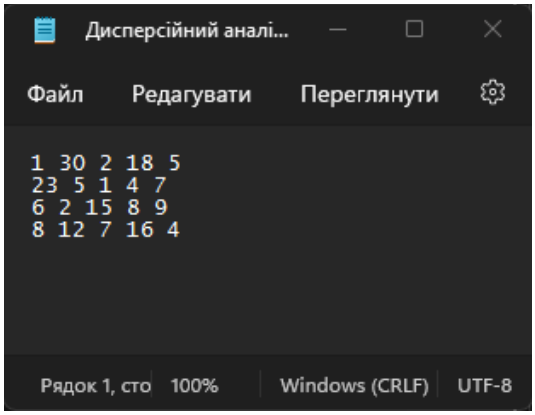


Рисунок 3.14 – Текстовий файл з прикладом

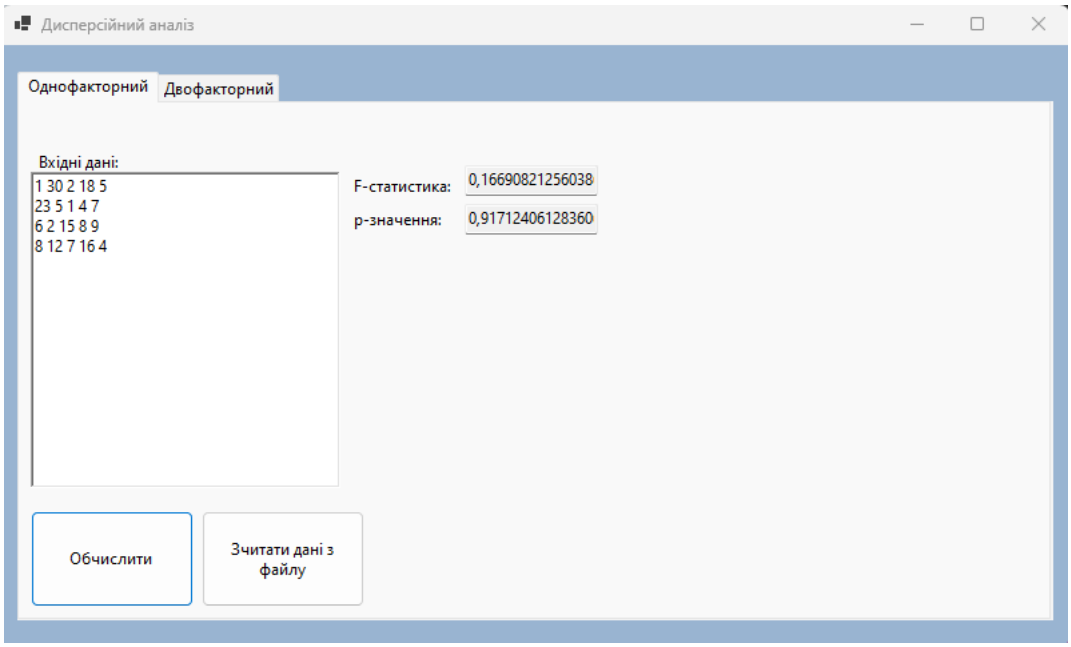


Рисунок 3.20 – Вікно рішення однофакторного дисперсійного аналізу з рішенням прикладу

3.4 Кореляційний, кластерний та дисперсійний аналізи

3.4.1 Кореляційний аналіз

Кореляція, або коефіцієнт кореляції, вимірює ступінь залежності між двома випадковими величинами. Зміна однієї або кількох з цих величин призводить до систематичних змін інших величин. Математично коефіцієнт кореляції служить мірою кореляції між випадковими величинами. Кореляція може бути позитивною або негативною, а також можливе відсутність статистичного зв'язку, наприклад, у випадку незалежних випадкових величин. Від'ємна кореляція вказує на те, що збільшення однієї змінної супроводжується зменшенням іншої змінної, а коефіцієнт кореляції є від'ємним. Позитивна кореляція означає, що збільшення однієї змінної супроводжується збільшенням іншої змінної та має позитивний коефіцієнт кореляції [11].

Кореляційний аналіз є методом обробки статистичних даних, який використовується для вивчення коефіцієнта кореляції між змінними. Цей метод включає порівняння коефіцієнтів між однією або багатьма парами ознак для виявлення статистичної взаємодії між ними.

Метою кореляційного аналізу є отримання інформації про одну змінну на основі іншої змінної. Якщо ця мета досягається, говорять про наявність кореляції між змінними. Загалом, гіпотеза про існування кореляції означає, що зміни значення змінної A збігаються з пропорційними змінами значення змінної B [11].

Варто зазначити, що кореляція відображає лише лінійну залежність між величинами і не відображає їх функціонального зв'язку.

Було обчислено коефіцієнти для пар метрика-експертна оцінка під час проведення кореляційного аналізу. Можна зробити такі висновки:

- Коефіцієнти кореляції для пар "метрика-експертна оцінка", які позначені червоним кольором, свідчать про наявність залежності між метриками та експертною оцінкою.

- Коефіцієнти кореляції для пар "метрика-експертна оцінка", позначені чорним кольором, не виявили значної залежності між ними (мала залежність).
- Значення коефіцієнтів кореляції не є дуже великими, що свідчить про те, що залежність між метриками та експертними оцінками, які досліджуються у даній курсовій роботі, не є значною.

Отримані дані від кореляційного аналізу будуть використані для проведення регресійного аналізу [11].

3.4.2 Кластерний аналіз

Кластерний аналіз, також відомий як аналіз кластерів або кластеризація даних, процес поділу певної вибірки об’єктів або ситуацій на групи, які називаються кластерами, з метою забезпечення подібності об'єктів всередині кожного кластера, тоді як об'єкти з різних кластерів суттєво відрізняються. Кластерний аналіз належить до галузі статистичної обробки та відноситься до навчання без учителя, охоплюючи широкий спектр завдань [12].

Кластерний аналіз є багатовимірною статистичною процедурою, яка збирає дані, що містять інформацію про вибірку об'єктів, і потім упорядковує ці об'єкти в однорідні групи, відомі як кластери (також відома як Q-кластеризація або Q-техніка). Кластер представляє собою групу елементів, які мають спільну властивість. Основною метою кластерного аналізу є знаходження груп схожих об'єктів у вибірці [12].

Кластерний аналіз має широкий спектр застосувань і використовується в різних галузях, таких як археологія, медицина, психологія, хімія, біологія, державне управління, філологія, антропологія, маркетинг, соціологія та інших наукових дисциплінах [12].

Однак, оскільки кластерний аналіз є універсальним, існує багато термінів, методів і підходів, які не обов’язково сумісні один з одним. Це ускладнює однозначне застосування та інтерпретацію кластерного аналізу з точки зору універсальності та послідовності [12].

Основні завдання кластерного аналізу включають:

- Розробка нової типології або класифікації на основі аналізу даних.
- Виявлення корисних концептуальних схем групування об'єктів.
- Формулювання гіпотез, що базуються на результатів аналізу даних.
- Перевірка гіпотез або дослідження, щоб визначити наявність типів (груп), які були виділені шляхом кластерного аналізу, у наявних даних.

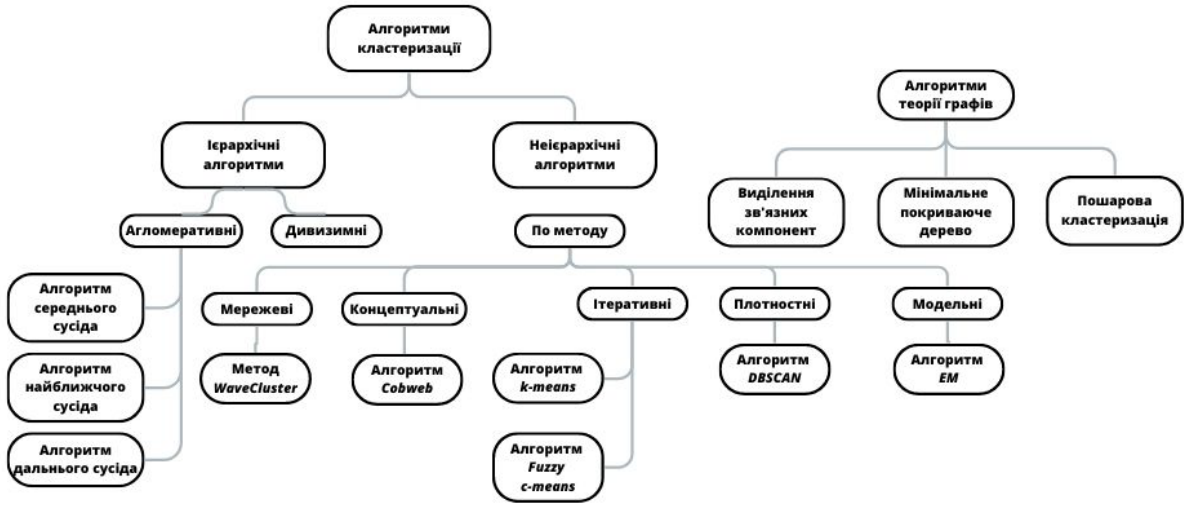


Рисунок 3.2.2 – Класифікація алгоритмів та методів кластерного аналізу

Незалежно від сфери дослідження, процес застосування кластерного аналізу включає такі етапи:

- Відбір підмножини об'єктів для проведення кластерного аналізу.
- Визначення і врахування багатьох змінних, за якими об'єкти у підмножині будуть оцінюватися.
- Обчислення значень міри схожості між об'єктами в підмножині.
- Застосування методу кластерного аналізу для групування схожих об'єктів.
- Перевірка достовірності результатів кластерного розподілення.

У кластерному аналізі встановлюються наступні вимоги до даних:

- Показники не повинні корелювати один з одним.
- Індикатор повинен бути безрозмірним.

- Розподіл показників повинен наближатися до нормального.
- Показники повинні бути "стійкими", тобто не піддаються випадковим впливам на їх значення.
- Вибірка повинна бути однорідною і не містити "викидів".

Якщо перед кластерним аналізом застосовується факторний аналіз, то вибірка не потребує коригування, оскільки вимоги автоматично виконуються процедурою факторного моделювання. Застосування z-стандартизації без негативних наслідків для вибірки є ще одним перевагою. В іншому випадку, вибірку потрібно коректувати [23].

3.4.3 Дисперсійний аналіз

Основною метою дисперсійного аналізу, фундаментальна концепція якого була запропонована Фішером у 1920 р., є дослідження значущості відмінності між середніми декількох груп даних або змінних. Якщо порівнюються середні двох груп, дисперсійний аналіз дасть той же результат, що і звичайний ґ-критерій для незалежних або залежних вибірок. Проте використання дисперсійного аналізу має переваги особливо для малих вибірок.

У дисперсійному аналізі перевірка статистичної значущості відмінності між середніми декількох груп здійснюється на основі вибіркових дисперсій. Ця перевірка проводиться за допомогою розбиття загальної дисперсії (варіації) на частини, одна з яких обумовлена випадковою помилкою (тобто внутрішньогруповою мінливістю), а друга пов'язана з відмінністю середніх значень. Якщо ця різниця є значною, нульова гіпотеза про те, що існує різниця між середніми значеннями, відхиляється за межі певного рівня значущості [24].

Однофакторний дисперсійний аналіз використовується для вивчення змін характеристик результатів під впливом зміни умов або рівнів факторів. Суть математичного перетворення дисперсійного методу полягає в порівнянні дисперсії, обумовленої фактором, з дисперсією всіх експериментально отриманих величин. Однофакторний аналіз вимагає принаймні трьох рівнів факторів і принаймні двох

випробувань на кожному рівні. Виконуючи дисперсійний аналіз, ми повинні переконатися, що немає різниці між нормальним розподілом досліджуваної випадкової змінної та дисперсією сукупності.

Двофакторний дисперсійний аналіз використовується для вивчення одночасного впливу двох факторів на різні вибірки об'єктів. Н. Коли різні зразки знаходяться під впливом різних комбінацій двох факторів. Одна змінна може мати істотний вплив лише на досліджувану властивість за наявності певних значень іншої змінної. Наприклад, підвищена мотивація прискорює вирішення проблем для людей з високим інтелектом, але уповільнює його для людей з низьким інтелектом. Тому двофакторний дисперсійний аналіз можна використовувати для оцінки не тільки впливу кожного фактора, але й їх взаємодії [24].

ВИСНОВКИ

1. У результаті даної роботи було створено програму для операційної системи Windows для підтримки підтримка лабораторного практикуму з дисципліни «Емпіричні методи програмної інженерії» за рахунок використання програмного забезпечення мовою C#.
2. Проведено аналіз аналогічних математичних пакетів та розроблено програмне забезпечення, спрямоване на цю конкретну тему. Також були зібрані статистичні дані щодо популярності розробки додатків для операційної системи Windows, що підтверджує актуальність даної теми. В роботі була надана інформація про використовувані системи під час процесу розробки. Було проведено дослідження основних складових додатку для операційної системи Windows. Вибраний та використаний необхідний інструментарій для розробки додатку, а також перераховані ключові бібліотеки та технології, які використовувалися.
3. Під час проектування програмного забезпечення було виконане моделювання різних компонентів, включаючи варіанти використання, класи, стани, артефакти та пакети.
4. На останньому етапі розробки було здійснене повне тестування додатку, включаючи перевірку всіх його компонентів. Цей процес пройшов успішно, оскільки проміжні тестування, які проводилися під час розробки програмного забезпечення, допомогли уникнути можливих проблем.
5. Розроблений додаток може застосовуватися студентами у процесі вивчення дисципліни «Емпіричні методи програмної дисципліни» спеціальності 121 Інженерія програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Емпіричні методи програмної інженерії. Методичні рекомендації до виконання розрахункової роботи на тему „Комп'ютерні технології статистичної обробки даних" для підготовки бакалаврів за спеціальністю 121 „Інженерія програмного забезпечення" / Укл. : Ющенко Н. Л. – Чернігів : ЧНТУ, 2018.
2. LEARN [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://learn.ztu.edu.ua/mod/page/view.php?id=9974&lang=ru
3. C# Language Documentation [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/
4. Empirical Research Methods in Software Engineering /Claes Wohlin, Martin Höst and Kennet Henningsson. – Sweden: Lund, Ronneby – 2003. – 1 page
5. Selecting Empirical Methods for Software Engineering Research [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://rumproarious.com/notes/selecting-empirical-methods-for-software-engineering-research/
6. Гарпуль О.З., Незамай Б.С. Лабораторний практикум «Емпіричні методи програмної інженерії» / О.З. Гарпуль, Б.С. Незамай. – Івано-Франківськ: Видавництво Прикарпатського національного університету, 2018.
7. Selecting Empirical Methods for Software Engineering Research / Steve Easterbrook, Janice Singer, Margaret-Anne Storey and Daniela Damian. – Sweden: Lund, – 2008. - Chapter 11
8. EZTESTS [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://www.eztests.xyz/criteria/anova_oneway/
9. UA-REFERAT [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://ua-referat.com/Дисперсійний_аналіз
10. INFORMATICS [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://informatics.in.ua/programming_csharp/part_01.php
11. PIDRU4NIKI [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://pidru4niki.com/14990528/ekonomika/korelyatsiyniy_analiz

12. PIDRU4NIKI [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://pidru4niki.com/11800912/ekonomika/klasterniy_analiz
13. ICSTUDIO [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://icstudio.online/post/etapi-zhittyevogo-ciklu-rozrobki-pz
14. Easterbrook, S., Singer, J., Storey, MA., Damian, D. (2008). Selecting Empirical Methods for Software Engineering Research. In: Shull, F., Singer, J., Sjøberg, D.I.K. (eds) Guide to Advanced Empirical Software Engineering. Springer, London. https://doi.org/10.1007/978-1-84800-044-5_11
15. Visual Studio Documentation [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://docs.microsoft.com/en-us/visualstudio
16. ROBOTDREAMS [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://robotdreams.cc/uk/blog/284-s-chto-eto-za-yazyk-i-gde-ego-ispolzuyut
17. STUDFILE [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://studfile.net/preview/5994722/page:7/
18. Мандель І. Д. Кластерний аналіз. - М.: Фінанси та статистика, 1988. ISBN 5-279-00050-7.
19. Accord.NET [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: http://accord-framework.net
20. Math.NET [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://numerics.mathdotnet.com
21. EVERGREENS [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://evergreens.com.ua/ua/articles/uml-diagrams.html
22. DOU [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://dou.ua/forums/topic/40575/
23. DOCS.KDE [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-elements.html
24. PIDRU4NIKI [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://pidru4niki.com/12560607/statistika/dispersiyniy_dvofaktorniy_analiz

25. Empirical Methods in Software Engineering Research / Walter F. Tichy and Frank Padberg. – Germany, 2007. – 2 page
26. LEARN MICROSOFT [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу: https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022
27. LEARN MICROSOFT [Електронний ресурс]: [Веб–сайт]. – Електронні дані. – Режим доступу:https://learn.microsoft.com/ru-ru/dotnet/framework/get-started/overview
28. Москалець О.В. Розробка програмного забезпечення для реалізації алгоритму кластерного аналізу методом ближнього сусіда: Матеріали студентської наукової конференції «Наука сьогодення: від досліджень до стратегічних рішень». 02.06.2023, ДУТ, м. Суми, 2023.
29. Москалець О.В Огляд дисципліни «Емпіричні методи програмної інженерії" і важливість практичних лабораторних занять / Москалець О.В // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії, 01-03 червня 2023р., ДУТ, м. Київ — К: ДУТ, 2023.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення для підтримки лабораторного практикуму з дисципліни "Емпіричні методи інженерії програмного забезпечення" мовою C#

Виконав студент 4 курсу

Групи ПД-44

Москалець Олександр Васильович

Керівник роботи

К.п.н, доцент кафедри ІПЗ Шевченко Світлана Миколаївна

Київ – 2023

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** — підтримка лабораторного практикуму з дисципліни «Емпіричні методи програмної інженерії» за рахунок використання програмного забезпечення мовою C#.
- **Об'єкт дослідження** — процес навчальної діяльності студентів при вивченні дисципліни «Емпіричні методи програмної інженерії».
- **Предмет дослідження** — програмне забезпечення для лабораторного практикуму з дисципліни «Емпіричні методи програмної інженерії».

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести огляд та аналіз тем дисципліни Емпіричні методи програмної інженерії.
2. Проаналізувати характеристики існуючих аналогів математичних пакетів з метою визначення їх можливостей для підтримки навчальної діяльності студентів.
3. Сформулювати вимоги до програмного забезпечення з урахуванням недоліків існуючих засобів.
4. Провести огляд та аналіз IT-засобів для розробки програмного забезпечення кваліфікаційної роботи бакалавра.
5. Спроектувати та розробити програмне забезпечення для розрахунку кореляційного та кластерного аналізу.
6. Провести тестування розробленого програмного забезпечення.

АНАЛІЗ АНАЛОГІВ МАТЕМАТИЧНИХ ПАКЕТІВ

Назва	Excel	MATLAB	MathCAD	Mathematica	Розроблений додаток
Платформи	Windows, Linux, MacOS, Android, IOS	Windows, Linux, MacOS	Windows	Windows, Linux, MacOS	Windows, Linux, MacOS
Видача завдання	Відсутня	Відсутня	Відсутня	Відсутня	Присутня
Пояснення завдання	Відсутня	Відсутня	Відсутня	Відсутня	Присутня
Розрахунок аналізу з текстового файлу	Відсутній	Відсутній	Відсутній	Присутній	Присутній
Підказки по ходу виконання завдання	Відсутня	Відсутня	Відсутня	Відсутня	Присутня
Збирання інформації в ході лабораторного практикуму	Відсутня	Відсутня	Відсутня	Відсутня	Присутня
Розрахунок аналізу	Присутній	Відсутній	Відсутній	Присутній	Присутній
Кросплатформеність	Присутня	Присутня	Відсутня	Присутня	Присутня

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

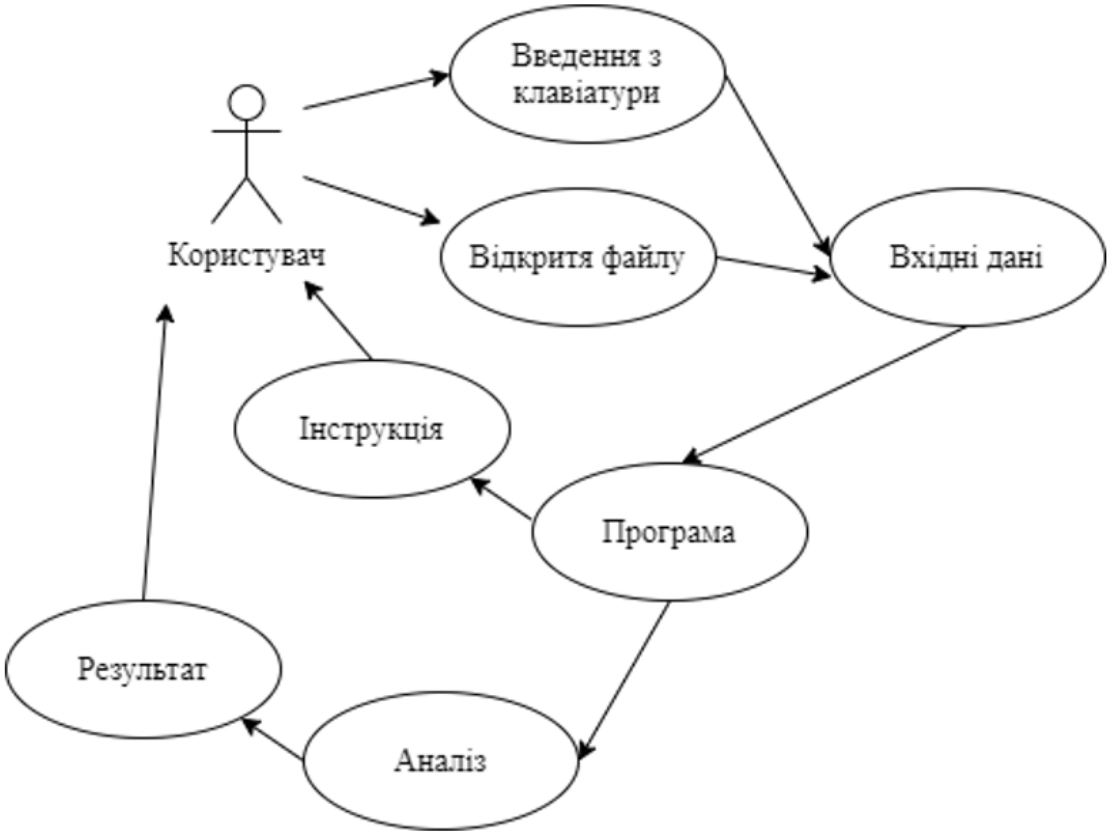
1. Відображення сторінки введення вхідних даних та результатів у процесі вирішення задач кореляційного, рангового, кластерного аналізу
2. Підказки по ходу виконання завдання кластерного та кореляційного аналізу.
3. Візуальне представлення виконання кластерного аналізу у вигляді дендрограми, кореляційного аналізу у вигляді діаграми розсіювання.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

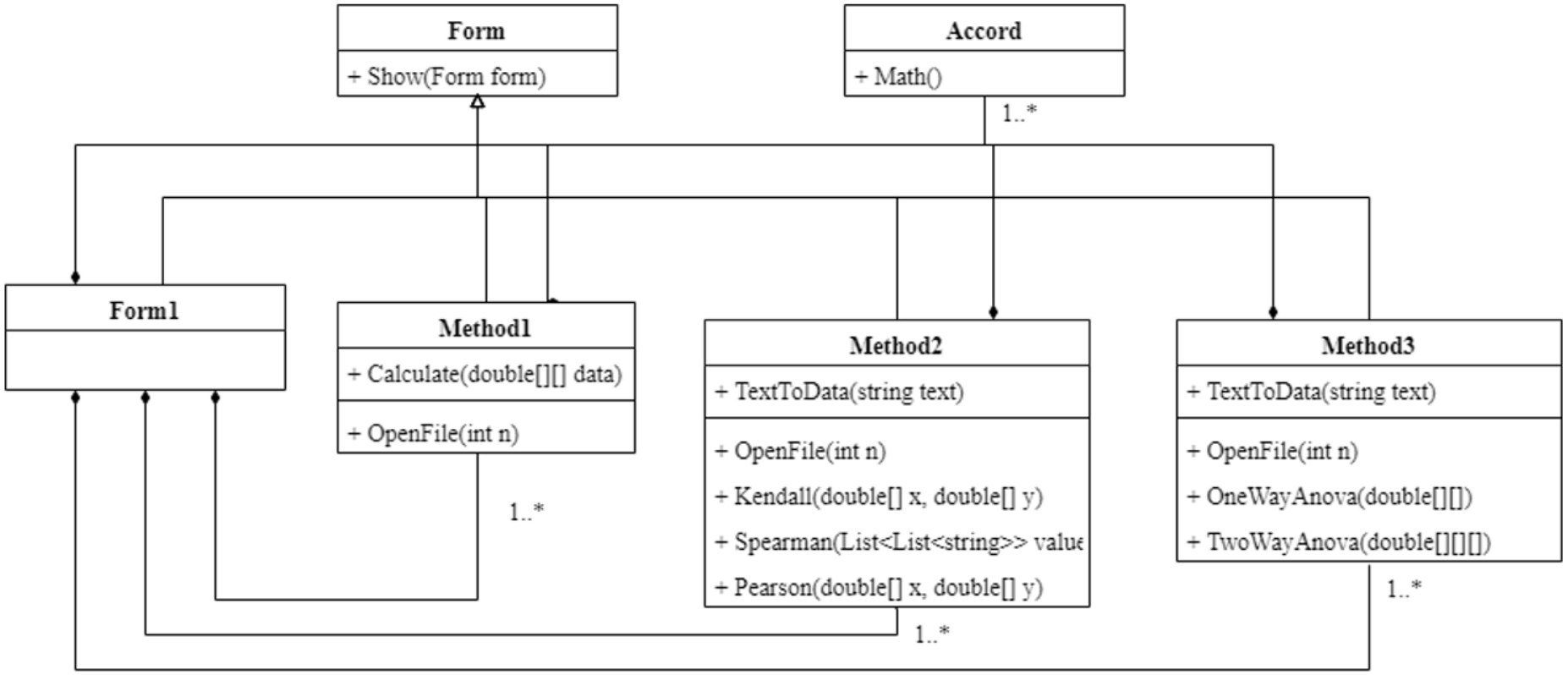
WinForm

Microsoft®
.NET

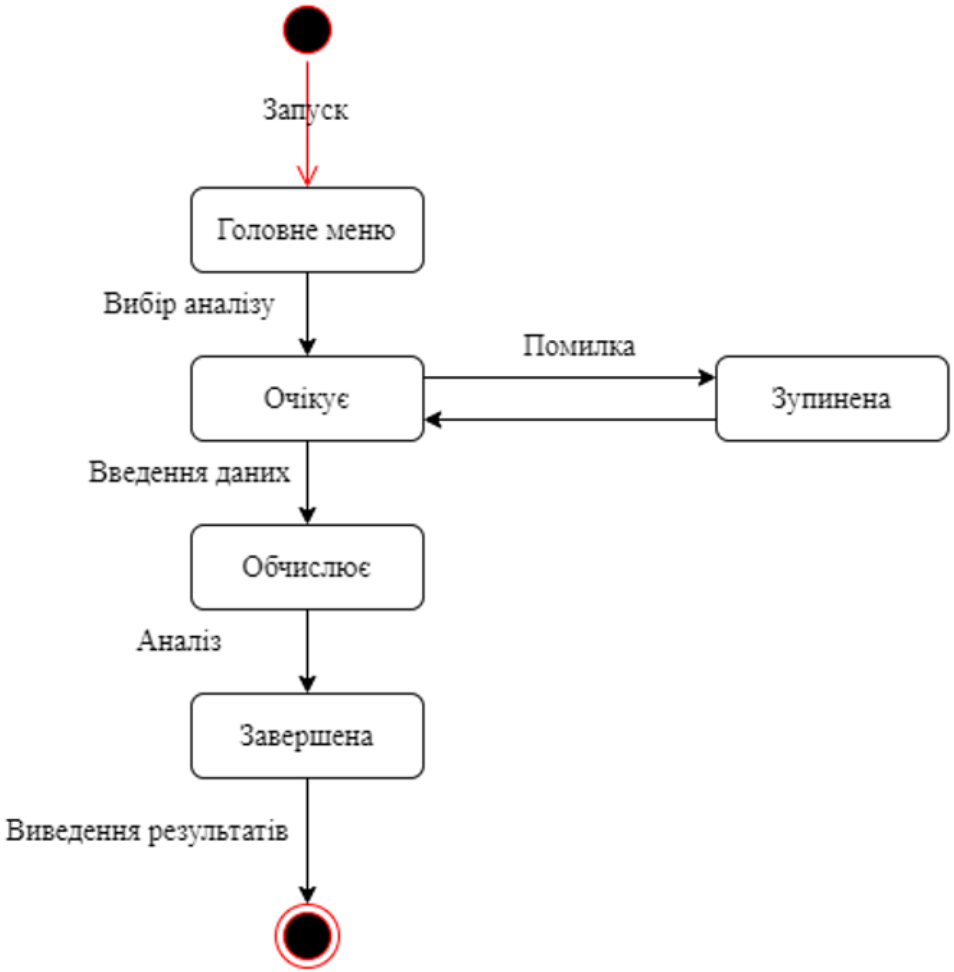
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



ДІАГРАМА КЛАСІВ

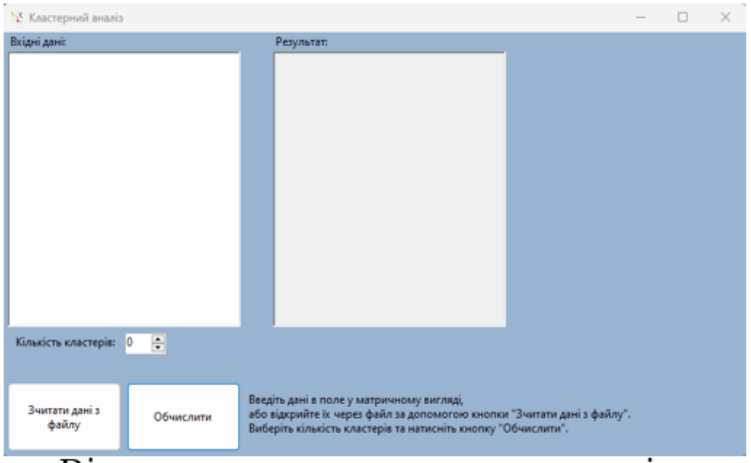


ДІАГРАМА СТАНІВ

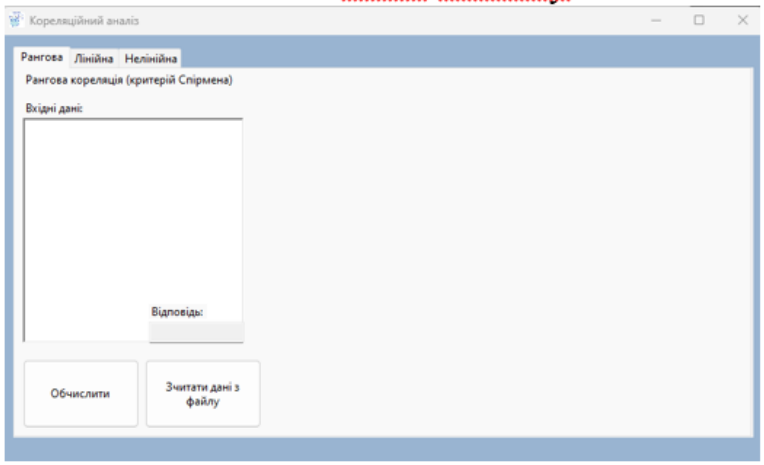


ЕКРАННІ ФОРМИ

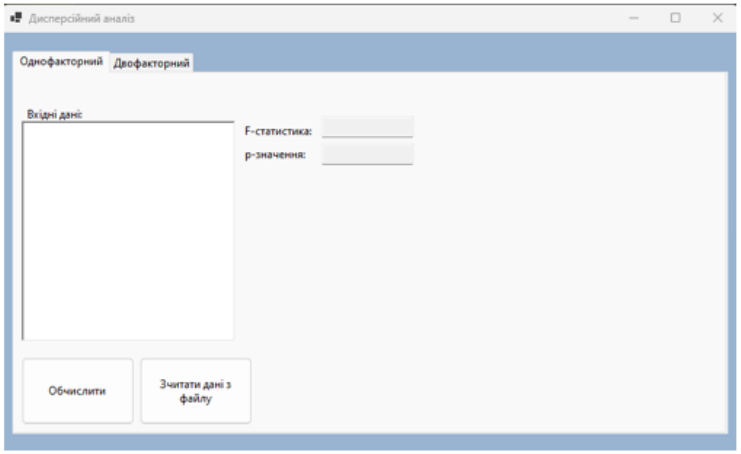
Головне вікно додатку



Вікно розрахунку кластерного аналізу

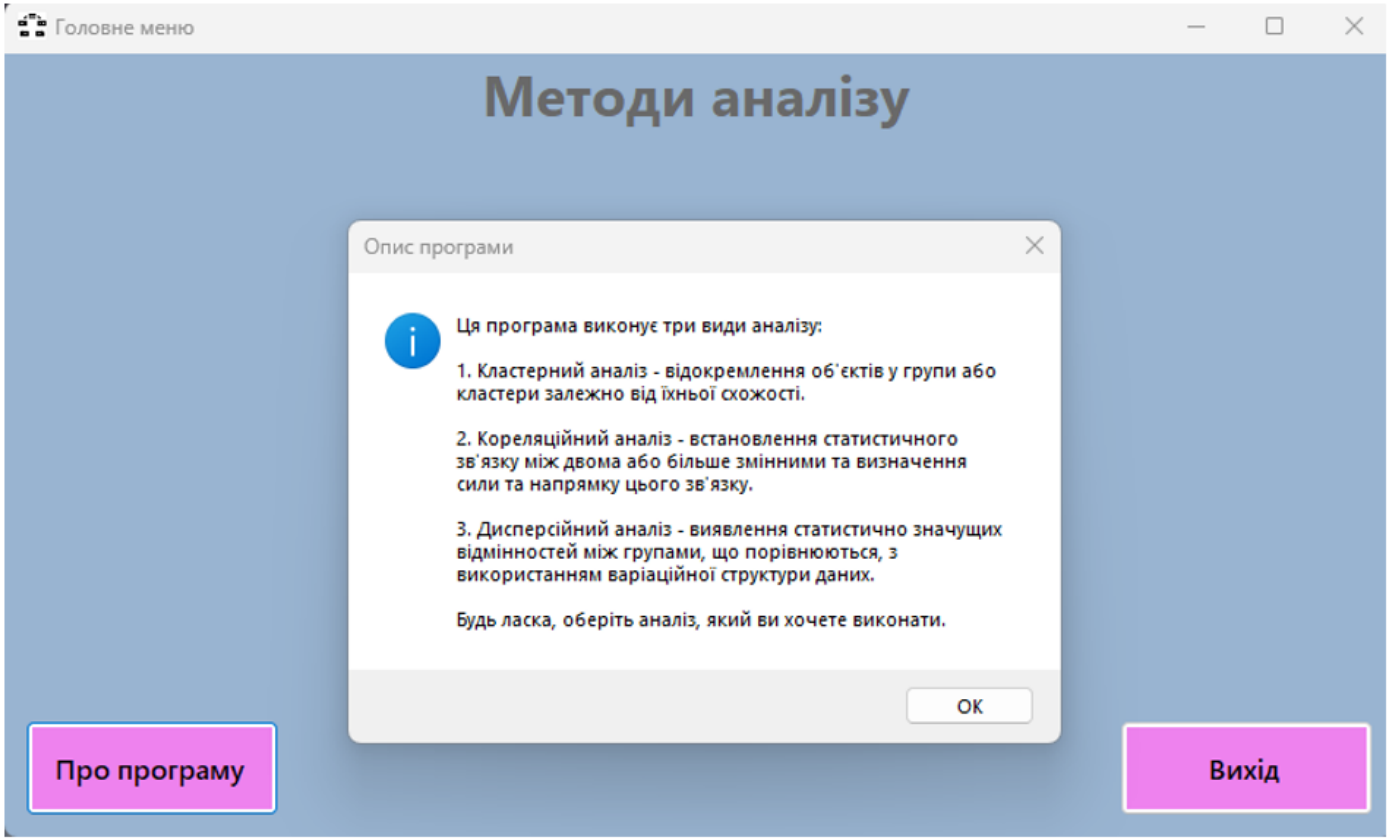


Вікно розрахунку кореляційного аналізу



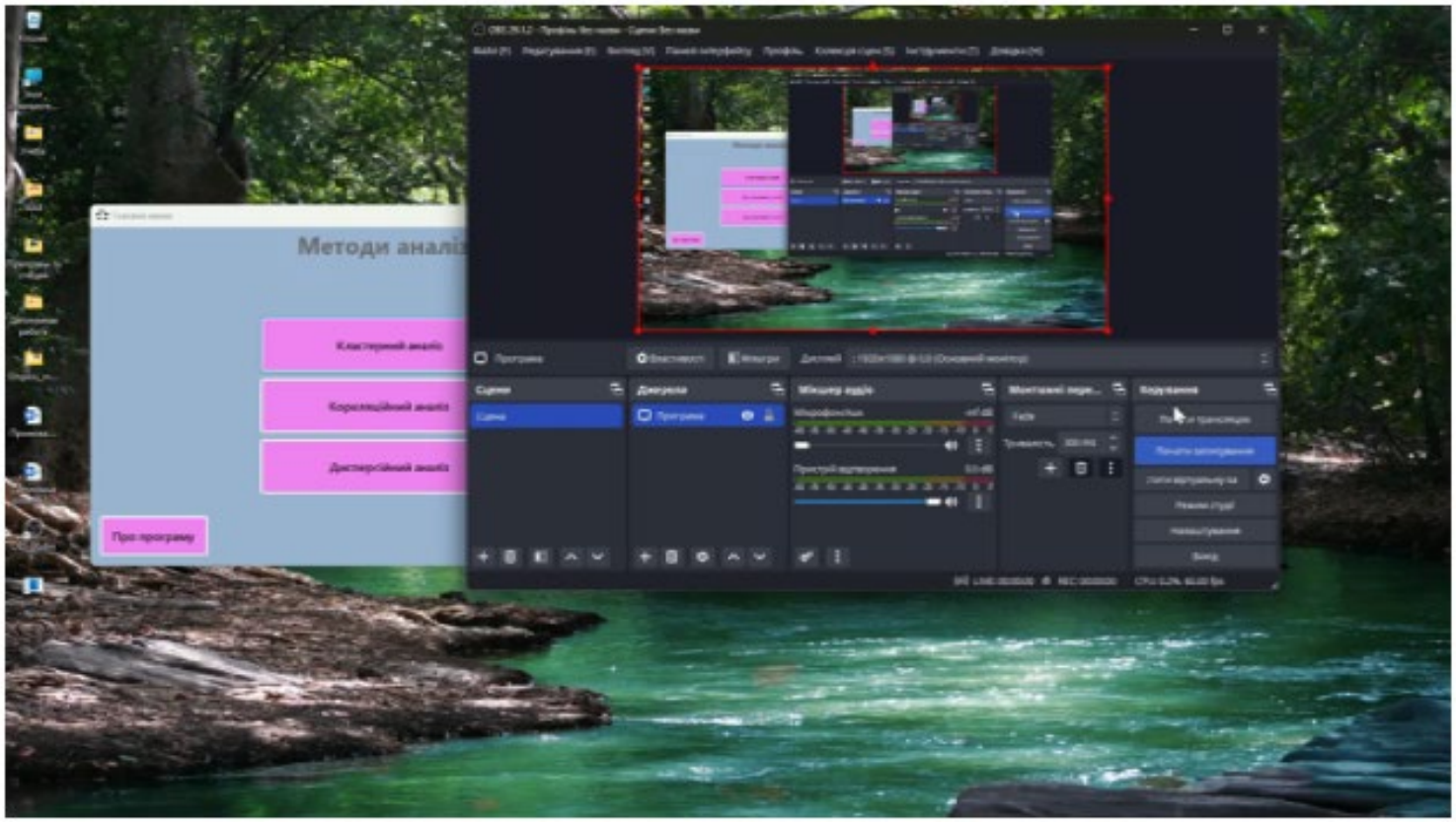
Вікно розрахунку дисперсійного аналізу

ЕКРАННІ ФОРМИ



Вікно “Опис програми”

Демонстрація додатку



АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Москалець О.В Розробка програмного забезпечення для реалізації алгоритму кластерного аналізу методом ближнього сусіда / Наука сьогодення: від досліджень до стратегічних рішень: матеріали VI Міжнародної студентської наукової конференції, м. Суми, 2.06.2023 / ГО «Молодіжна наукова ліга». — Вінниця: ГО «Європейська наукова платформа», 2023. – с. 142 - 143
2. Москалець О.В Огляд дисципліни «Емпіричні методи програмної інженерії" і важливість практичних лабораторних занять / Москалець О.В // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії, 01-03 червня 2023р., ДУТ, м. Київ — К: ДУТ, 2023. Подано до друку.

ВИСНОВКИ

1. Проведено аналіз існуючих математичних пакетів, таких як Microsoft Excel, MATLAB, MathCAD, Wolfram Mathematica. Більшість програм носять обчислювальний характер. Ключовим недоліком є відсутність поетапної підтримки у використанні алгоритмів аналізу даних.
2. Проведено аналіз засобів розробки програмного забезпечення, обгрунтовано їх вибір. Додаток реалізований в інтегрованому середовищі розробки Microsoft Visual Studio, використовуючи інтерфейсове програмування додатків Windows Form, мовою програмування C#.
3. Розроблено програмне забеспечення, яке надає можливість підтримати навчальний процес з дисципліни «Емпіричні методи програмної інженерії» за рахунок використання програмного забезпечення мовою C#. Також розроблений додаток може бути використано у всіх сферах, де використовується кореляційний аналіз (лінійна, рангова та нелінійна кореляція) та кластерний аналіз.