МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

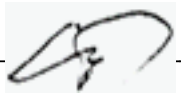
Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_  _ проф. Приходько С.Б.

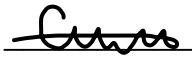
«___» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка програми для оцінювання метрики LOC програмних застосунків на C++

Виконав: студент групи 4151

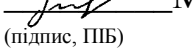
 Сімонов Г.В.

(підпис, ПІБ)

Керівник роботи:

доцент кафедри ПЗАС, к.т.н., доцент

(посада, науковий ступень вчене звання)

 Макарова Л. М

(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
«_08_» ___04___ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту_____Сімонову Герману Вячеславовичу_____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програми для оцінювання метрики LOC програмних застосунків на C++_____

Керівник роботи_____доцент кафедри ПЗАС, к.т.н., доцент Макарова Л. М_____

Затверджені наказом ректора №153 від 08.04.2025 р.

2. Термін подання роботи: 02.06.2025 р. _____

3. Вихідні дані по роботі:_____

4. Перелік питань, що належать до розробки (найменування розділів)_____

Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів _____

Титульний слайд, Мета та завдання кваліфікаційної роботи, Постановка задачі на розробку ПЗ, Аналіз вимог до ПЗ, Діаграми діяльності ПЗ, Архітектура ПЗ, Діаграма класів ПЗ, Діаграми послідовностей ПЗ, Програмні засоби розробки ПЗ, Результати розробки ПЗ, Висновки, Заключний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 08.04.2025 р.

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	31.03.2025	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	07.04.2025	ПП*
3.	Аналіз вимог до ПЗ	14.04.2025	ПП*
4.	Розробка архітектури ПЗ	21.04.2025	
5.	Розробка проєкту ПЗ	05.05.2025	
6.	Реалізація та тестування ПЗ	12.05.2025	
7.	Підготовка розділу Результати розробки ПЗ	16.05.2025	
8.	Підготовка розділу з охорони праці	19.05.2025	ПП*
9.	Підготовка розділу ВИСНОВКИ	23.05.2025	
10.	Оформлення списку використаних джерел та додатків	26.05.2025	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	02.06.2025	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
12.	Підготовка презентації та доповіді	06.06.2025	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	09.06.2025	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	16.06.2025	

* - за результатами переддипломної практики (ПП), яка була з 03.02.2025 до 23.03.2025 р.

Студент _____ Сімонов Г.В.
(підпис) (ПІБ)

Керівник роботи _____ Макарова Л.М.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розв’язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці програми для оцінювання метрики LOC програмних застосунків на C++, що дозволить користувачам швидко отримувати прогнозне значення метрики LOC C++-проєктів.

Кваліфікаційна робота містить аналіз та опис предметної галузі за темою кваліфікаційної роботи, аналіз існуючих аналогів та постанова задачі на розробку програми. Розроблено архітектуру, ескізний, технічний та робочий проєкти програми, представлено результати розробки та розділ з охорони праці.

Кваліфікаційна робота викладена на 77 сторінках машинописного тексту та містить: 12 рисунків, 10 таблиць, 5 додатків, список використаних джерел з 13 найменувань.

ABSTRACT

The qualification work is dedicated to solving the practical task of software engineering, characterized by complexity and uncertainty of conditions, the software development for estimating the LOC metric of software applications in C++, which will allow users to quickly obtain the predictive value of the LOC metric of C++ projects.

In qualification work includes subject area analysis and description based on the qualification work thesis, analysis of existing analogues and formulation of the problem for software development. The architecture, sketch, technical and working projects of the software, results of development and the section on labor protection are developed.

The qualification work is presented on 77 pages of typewritten text and contains: 12 figures, 10 tables, 5 appendices, list of references from 13 names.

ЗМІСТ

ВСТУП

У сучасному програмному інжинірингу надзвичайно важливим є ефективний контроль якості та складності програмного забезпечення, що розробляється. Із розвитком інформаційних технологій зростає складність програмних продуктів, а також збільшується обсяг коду, який необхідно супроводжувати, тестувати та підтримувати. У зв’язку з цим зростає потреба у застосуванні формалізованих методів оцінювання характеристик програмного продукту. Одним із таких методів є використання кількісних метрик, які дозволяють об'єктивно вимірювати певні властивості програмного коду [1]. Такі метрики можуть включати кількість помилок, покриття тестами, час виконання, складність алгоритмів, а також кількість рядків коду (LOC - Lines of Code). Метрика LOC є однією з найпростіших, але водночас дуже важливою, оскільки вона дає змогу оцінити розмір коду, його зростання в динаміці, а також використовується як вихідна інформація для обчислення інших метрик, таких як продуктивність розробки (кількість LOC на день) чи технічний борг.

Метрика LOC має особливе значення під час аналізу вихідного коду великих проєктів, особливо коли необхідно оцінити ефективність команди, планувати подальші етапи розробки або приймати рішення щодо оптимізації існуючого коду. Водночас її використання вимагає точного та об’єктивного підходу до підрахунку, з урахуванням синтаксичних особливостей мови програмування, на якій написано код.

Мова програмування C++ продовжує залишатися популярною для створення високопродуктивних систем, вбудованого програмного забезпечення та застосунків з високими вимогами до ефективності. Вона залишається в трійці лідерів, утримуючи друге місце в рейтингу TIOBE Index протягом останнього року, а у 2022 році навіть стала мовою програмування року, як мова програмування, яка мала найвищий ріст рейтингів за рік [2]. У зв’язку з цим актуальним завданням є створення інструментів, що дають змогу автоматизувати

оцінювання метрик C++-проєктів, зокрема LOC, з урахуванням особливостей синтаксису цієї мови.

Кваліфікаційна робота передбачає розв’язання практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов – розробку програми для оцінювання метрики LOC програмних застосунків на C++, що дозволить користувачам швидко отримувати прогнозне значення метрики LOC C++-проєктів, що, в свою чергу, може бути використано для оцінки продуктивності команд розробників, технічного боргу, а також при порівнянні різних версій одного й того самого програмного забезпечення.

Метою роботи є розробка програми для оцінювання метрики LOC програмних застосунків на C++.

Для досягнення поставленої мети необхідно:

– провести аналіз практичної задачі інженерії програмного забезпечення, яка стосується вказаної мети;

– проаналізувати існуючі аналоги та обґрунтувати необхідність розробки програми для оцінювання метрики LOC програмних застосунків на C++;

– здійснити постановку задачі на розробку програми для оцінювання метрики LOC програмних застосунків на C++;

– розробити архітектуру програми для оцінювання метрики LOC програмних застосунків на C++;

– розробити ескізний, технічний та робочий проєкти програми для оцінювання метрики LOC програмних застосунків на C++;

– здійснити кодування, тестування та випробування розробленої програми для оцінювання метрики LOC програмних застосунків на C++;

– розробити усю необхідну програмну документацію.

Об'єктом кваліфікаційної роботи є процес розробки програми для оцінювання метрики LOC програмних застосунків на C++.

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ МЕТРИКИ LOC ПРОГРАМНИХ ЗАСТОСУНКІВ НА C++

1.1 Аналіз практичної задачі інженерії програмного забезпечення

При аналізі практичної задачі інженерії програмного забезпечення, яку має розв’язувати дана кваліфікаційна робота, а саме розробці програми для оцінювання метрики LOC програмних застосунків на C++, для забезпечення максимальної ефективності від експлуатації програми важливо враховувати всі особливості предметної галузі функціонування програмного продукту.

В інженерії програмного забезпечення одним з ключових аспектів є прогнозування трудомісткості, обсягів коду та ресурсів, необхідних для реалізації програмного продукту. Таке прогнозування дає змогу ефективно планувати час розробки, розподіляти завдання між членами команди, оцінювати складність майбутнього проєкту ще на етапі його проєктування. У цьому контексті метрика LOC виступає як одна з важливих метрик, значення якої використовується для приблизної оцінки розміру програмного забезпечення та визначення витрат на його розробку й супровід, особливо на початкових стадіях проєктування.

Однак сам по собі підрахунок значення LOC є лише частиною загального процесу оцінювання складності проєкту. У практичній розробці часто виникає потреба у попередньому прогнозуванні кількості рядків коду на основі початкових характеристик проєкту, таких як кількість модулів, класів, функцій тощо. Одним з перспективних підходів у цьому напрямку є використання регресійного моделювання, яке дозволяє встановити математичну залежність між кількістю класів у програмі та очікуваною кількістю рядків коду. Таке моделювання потребує попереднього аналізу даних із реальних програмних проєктів, на основі яких будується модель, здатна робити обґрунтовані прогнози для нових програмних продуктів.

Практична задача, яка розглядається в межах цієї кваліфікаційної роботи, полягає у створенні програми, яка на основі навчальної вибірки з існуючих C++-проєктів будує математичну залежність між кількістю класів та кількістю LOC, і використовує її для прогнозування LOC у нових проєктах.

Це завдання передбачає кілька етапів:

- формування навчальної вибірки з уже існуючих програмних застосунків на C++;

- реалізацію програмного інтерфейсу для побудови відповідної регресійної моделі (нелінійної) та прогнозування на її основі очікуваної кількості LOC;

- порівняння прогнозного значення з фактичним (якщо воно доступне) для перевірки точності моделі.

У результаті реалізації такого інструменту розробники зможуть отримувати не лише фактичну статистику про вихідний код, а й попередні оцінки обсягів коду на етапі проєктування, що робить цей підхід особливо корисним для проєктного планування, оцінки вартості, керування ризиками та прийняття рішень на ранніх етапах життєвого циклу програмного забезпечення. Таким чином, запропоноване рішення спрямоване на автоматизацію та інтелектуалізацію практичних задач сучасної інженерії програмного забезпечення.

1.2 Програмні інструменти для побудови моделей

Вузькоспеціалізованого автоматизованого існуючого програмного забезпечення для оцінювання метрики LOC програмних застосунків на C++ знайдено не було, тож розглянемо інструменти, за допомогою яких можливо в поєднанні з ручним доопрацюванням вирішити поставлену задачу.

MATLAB. Професійне середовище для обробки числових даних, з великою бібліотекою статистичних інструментів. MATLAB поєднує у собі середовище робочого столу, налаштоване для ітераційного аналізу та процесів проєктування, з мовою програмування, яка безпосередньо виражає математику матриць і масивів

[3]. Комерційний інструмент. Для вирішення поставленної задачі потребує написання відповідної програми, але фахівцю без підготовки, наприклад. менеджеру без досвіду програмування, працювати з ним буде складно.

Excel з пакетом «Аналіз даних». Підходить для простих моделей, коли обсяг даних невеликий. Дає змогу будувати трендові лінії та отримувати коефіцієнт кореляції [4]. Однак при необхідності побудови нелінійних моделей, які використовують нормалізацію даних, вимагає великої кількості ручних операцій, що може призвести до значної кількості помилок.

CodeScene. Комерційний інструмент, який на основі історії репозиторію і метрик коду намагається прогнозувати проблемні ділянки, технічний борг тощо. Частково використовує машинне навчання [5]. Працює з проєктами в одному репозиторії. Не дозволяє напряму будувати регресійні моделі та отримувати прогнозне значення метрики LOC.

Тому було прийнято рішення про розробку власного програмного забезпечення - програми для оцінювання метрики LOC програмних застосунків на C++.

Для вирішення поставленої задачі доцільно застосувати елементи регресійного аналізу. Найпростіший варіант – лінійна регресія, яка використовується в статистиці, залежність однієї змінної від іншої (або кількох інших змінних) з лінійною функцією залежності. Але лінійну регресію можна застосовувати у разі, коли випадкові змінні розподілені за нормальним законом розподілу, що на практиці зустрічається не завжди. Тож для зменшення помилок та підвищення достовірності розрахунків потрібно використовувати нелінійну регресію.

Для підвищення точності побудови рівняння регресії намагаються знайти його довірчий інтервал. У випадку нормального закону розподілу випадкових величин довірчий інтервал лінійного рівняння регресії можливо побудувати традиційним методом з використанням t-розподілу Стьюдента. Аналогічно також можна побудувати інтервал прогнозування лінійного рівняння регресії. Для

перевірки якості отриманої моделі використовують такі критерії, як: R^2, *MMRE*, *PRED*(0,25) [6].

На даному етапі аналіз практичної задачі інженерії програмного забезпечення, яку має розв'язувати кваліфікаційна робота, закінчуємо, та переходимо до постановки задачі на розробку програми для оцінювання метрики LOC програмних застосунків на C++.

1.3 Постановка задачі на розробку програми для оцінювання метрики LOC програмних застосунків на C++

Аналіз предметної області та існуючого програмного забезпечення, показав необхідність розробки програми для оцінювання метрики LOC програмних застосунків на C++. Розроблювана програма повинна надавати своїм користувачам наступні можливості:

розрахувати числові характеристики вибірки, такі, як: кількість значень, вибіркове середнє, дисперсія, асиметрія, ексцес, результат тесту Пірсона на нормальність розподілу випадкової величини;

побудувати гістограми вихідної вибірки та графік з її точками;

– розрахувати числові характеристики регресії, такі, як: коефіцієнти рівняння регресії (b_0 та b_1), коефіцієнт детермінації (R_2), середнє значення відносної похибки (*MMRE*), відсоток значень, які мають відносну похибку 0,25 (*PRED*(0,25));

розрахувати значення границь довірчого інтервалу та інтервалу прогнозування;

побудувати графік регресії, довірчого інтервалу та інтервалу прогнозування;

розрахувати значення метрики LOC за введеним значенням кількості класів;

переглянути точки отриманої вибірки, представлені числовими значеннями;

переглянути розраховані точки регресії, довірчого інтервалу та інтервалу прогнозування у вигляді числових значень.

Для забезпечення можливості виконання вище наведених функцій програма повинна приймати вхідні данні з структурою наведеною нижче.

Вхідні дані:

для розрахунку регресії: кількість класів, LOC;

для прогнозування значення метрики LOC згідно регресії: кількість класів.

Для розрахунку регресії введення даних здійснюється із файлу формату .csv, який містить наступні стовпці: значення кількості класів NC, значення метрики LOC.

Для прогнозування значення метрики LOC згідно регресії введення даних здійснюється до відповідного поля вводу вручну з клавіатури.

Відповідно до складу виконуваних функції програма повинна надавати користувачу можливість перегляду інформації з структурою наведеною нижче.

Інформація про регресію – сторінка, яка містить інформацію:

графік, який відображає лінію регресійної функції, довірчий інтервал, інтервал прогнозування, точки даних;

числові характеристики регресії, такі, як: коефіцієнти рівняння регресії (b_0 та b_1), коефіцієнт детермінації (R_2), середнє значення відносної похибки (*MMRE*), відсоток значень, які мають відносну похибку 0,25 (*PRED*(0,25));

значення метрики LOC – відображення на графіку моделі розрахованої точки.

Інформація про вихідну вибірку – сторінка, яка містить інформацію:

гістограми вибірки;

числові характеристики вибірки, такі, як: кількість значень, вибіркове середнє, дисперсія, асиметрія, ексцес, результат тесту Пірсона на нормальність розподілу випадкової величини;

відповідні точки вибірки.

Більш докладно вимоги до програмного забезпечення наведено в Додатку А – Технічне завдання.

2 РОЗРОБКА ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ МЕТРИКИ LOC ПРОГРАМНИХ ЗАСТОСУНКІВ НА C++

2.1 Аналіз вимог до програми для оцінювання метрики LOC програмних застосунків на C++

2.1.1 Побудова моделі варіантів використання

Аналіз вимог до програми для оцінювання метрики LOC програмних застосунків на C++ розпочнемо з виявлення варіантів використання.

На даному етапі розробки програми були виявлені варіанти використання, які представлені в табл. 2.1. Програма передбачає наявність одного актора – користувача програми.

Таблиця 2.1 – Варіанти використання програми для оцінювання метрики LOC програмних застосунків на C++

Найменування варіанту використання	Формулювання варіанту використання
1	2
Розрахувати характеристики вибірки	Розраховує числові характеристики вибірки такі, як: кількість значень, вибіркове середнє, дисперсія, асиметрія, ексцес, результат тесту Пірсона на нормальність розподілу випадкової величини, а також відображає гістограми розподілу випадкової величини та відповідні точки на графіку.
Розрахувати регресію	Розраховує коефіцієнти рівняння регресії, довірчий інтервал і інтервал прогнозування, та відображає результат у вигляді графіку, на якому зображені відповідні лінії та точки.

Продовження табл. 2.1

1	2
Отримати дані	Отримує дані із файлу формату .csv, який містить наступні стовпці: значення кількості класів NC, значення метрики LOC
Переглянути точки вихідної вибірки у вигляді списку чисел	Відображає координати відповідних точок вибірки у вигляді списку значень.
Переглянути розраховану регресію у вигляді списку чисел	Відображає результат розрахунку у вигляді списку значень (координат точок, значень границь довірчого інтервалу та інтервалу прогнозування).
Розрахувати значення метрики LOC	Розраховує значення метрики LOC в залежності від введеного значення кількості класів та відображає результат у вигляді числа, також подається значення границь довірчого інтервалу та інтервалу прогнозування.

Діаграма варіантів використання програми для оцінювання метрики LOC програмних застосунків на C++ представлена на рис. 2.1.

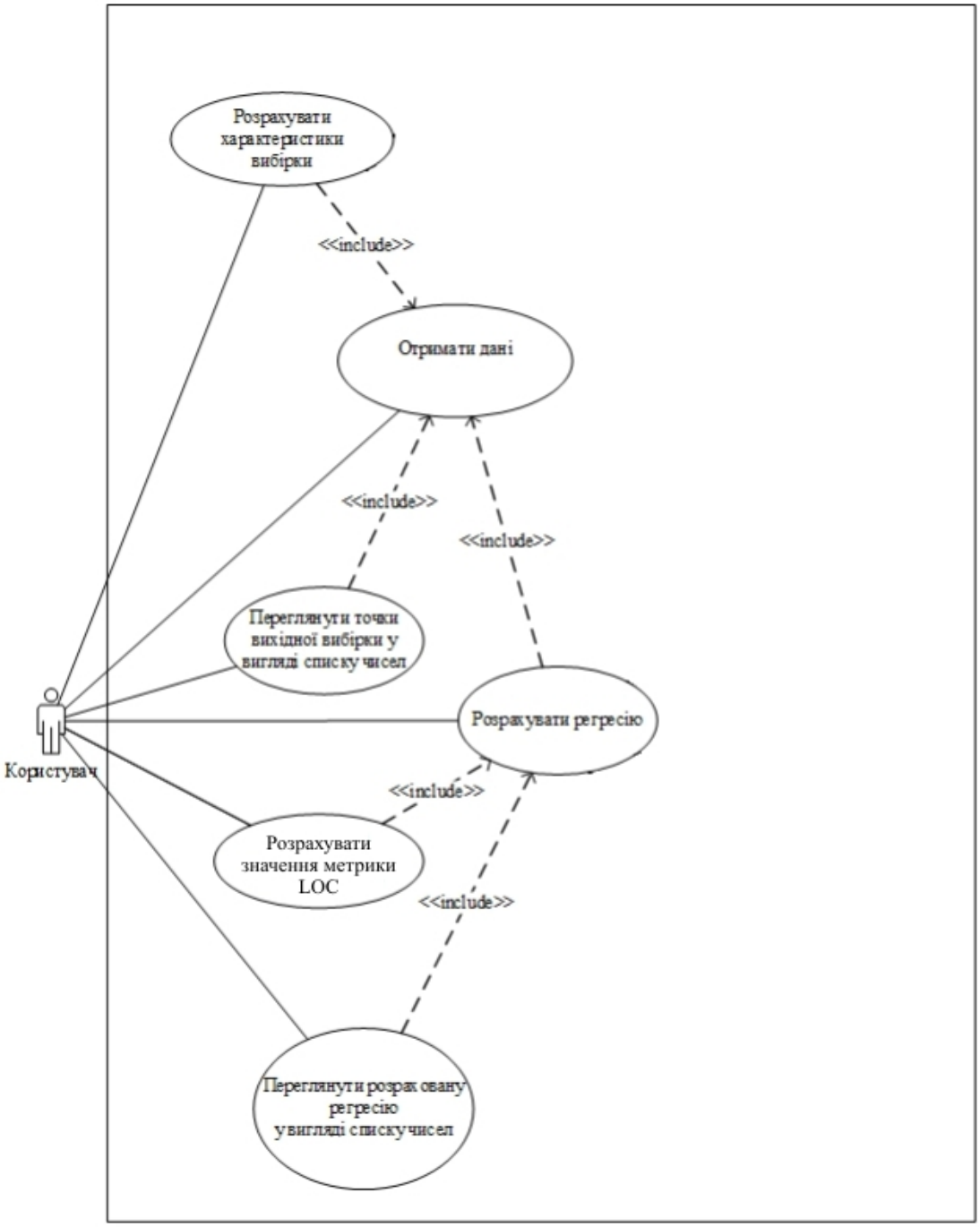


Рисунок 2.1 – Діаграма варіантів використання програми для оцінювання метрики LOC програмних застосунків на C++

Далі переходимо до розробки специфікації варіантів використання програми для оцінювання метрики LOC програмних застосунків на C++.

2.1.2 Специфікації варіантів використання

Були розроблені специфікації варіантів використання програми для оцінювання метрики LOC програмних застосунків на C++. Специфікації варіантів використання наведені у таблицях 2.2 – 2.7.

Таблиця 2.2 – Специфікація варіанту використання "Розрахувати характеристики вибірки"

Складова	Опис
Опис прецеденту	Даний варіант використання ініціює користувач. Розраховує такі числові характеристики вибірки, як: кількість даних, мінімальне та максимальне значення, дисперсія, асиметрія, результат тесту Пірсона на нормальність розподілу. Відображає на одній сторінці (вкладці) гістограми розподілу випадкової величини, розраховані числові характеристики у вигляді таблиць, а також відображає відповідні точки вибірки на графіку.
Передумови	Перед активізацією варіанта використання повинен бути виконаний потік варіанта використання «Отримати дані»
Базовий потік	Користувач обирає варіант розрахунку характеристик вихідної вибірки. Програма розраховує вищезазначені числові характеристики та одразу відображає сторінку (вкладку) з числовими характеристиками вихідної вибірки у вигляді таблиць і гістограмами розподілу випадкової величини, а також графік з відповідними точками.
Альтернативні потоки	-
Постумови	Відсутні

Таблиця 2.3 – Специфікація варіанту використання "Розрахувати регресію"

Складова	Опис
Опис прецеденту	Даний варіант використання ініціює користувач. Розраховує коефіцієнти рівняння регресії, границі довірчого інтервалу і інтервалу прогнозування, та відображає результат у вигляді відповідних ліній на графіку, а також відображає числові характеристики, такі як: коефіцієнти рівняння регресії (b_0 та b_1), коефіцієнт детермінації (R_2), середнє значення відносної похибки (*MMRE*), відсоток значень, які мають відносну похибку 0,25 (*PRED*(0,25)).
Передумови	Перед активізацією варіанта використання повинен бути виконаний потік варіанта використання «Отримати дані»
Базовий потік	Користувач обирає «Розрахувати регресію» за допомогою методу нормалізації десятковим логарифмом. Програма проводить необхідні розрахунки та відображає результат у вигляді відповідних ліній на графіку, а також відображає числові характеристики регресії.
Альтернативні потоки	Нормалізація вихідної вибірки не спрацювала. Якщо нормалізація вихідної вибірки обраним методом не можлива, немає сенсу продовжувати розрахунок регресії. Програма відображає повідомлення про помилку та зупиняє виконання даного варіанту використання.
Постумови	Відсутні

Таблиця 2.4 – Специфікація варіанту використання "Переглянути точки вихідної вибірки у вигляді списку чисел"

Складова	Опис
Опис прецеденту	Даний варіант використання ініціює користувач. Відображає координати відповідних точок вибірки, що була отримана. Відображення подається у вигляді таблиці, стовбцями є: номер точки в масиві точок, кількість класів та значення метрики LOC.
Передумови	Перед виконанням базового потоку варіанта використання повинен бути виконаний потік варіанта використання «Отримати дані»
Базовий потік	Користувач обирає варіант використання переглянути точки вихідної вибірки у вигляді списку чисел. Програма відображає список числових значень отриманої вибірки у вигляді таблиці.
Альтернативні потоки	Дані не отримані. Програма пропонує виконати потік «Отримати дані» повторно
Постумови	Відсутні

Таблиця 2.5 – Специфікація варіанту використання "Переглянути розраховану регресію у вигляді списку чисел"

Складова	Опис
Опис прецеденту	Даний варіант використання ініціює користувач. Відображає результат розрахунку регресії у вигляді списку значень (координат точок регресійної функції, значень границь довірчого інтервалу та інтервалу прогнозування) у вигляді таблиці.
Передумови	Перед активізацією варіанта використання повинен бути виконаний потік варіанта використання «Отримати дані»
Базовий потік	Користувач обирає переглянути розраховану регресію у вигляді списку чисел; Програма відображає відповідний список числових значень у вигляді таблиці.
Альтернативні потоки	Відсутні
Постумови	Відсутні

Таблиця 2.6 – Специфікація варіанту використання "Розрахувати значення метрики LOC"

Складова	Опис
Опис прецеденту	Даний варіант використання ініціює користувач. Розраховує значення метрики LOC в залежності від введеного значення кількості класів та відображає результат у вигляді числа, також подається значення довірчого інтервалу та інтервалу прогнозування.
Передумови	Перед активізацією варіанта використання повинен бути виконаний потік варіанта використання «Розрахувати регресію»
Базовий потік	Користувач вводить значення кількості класів. Програма проводить необхідний розрахунок та відображає результат у вигляді числа, також подається значення довірчого інтервалу та інтервалу прогнозування. Крім цього, на графіку регресійної функції та на лініях інтервалів прогнозування та довірчого, програма відображає відповідні розраховані точки.
Альтернативні потоки	Введене значення кількості класів не знаходиться в припустимих межах. Програма відображає повідомлення про помилку та зупиняє виконання даного варіанту використання.
Постумови	Відсутні

Таблиця 2.7 – Специфікація варіанту використання "Отримати дані"

Складова	Опис
Опис прецеденту	Даний варіант використання ініціює користувач. Отримає дані з файлу .csv
Передумови	Відсутні
Базовий потік	Користувач обирає варіант використання отримати дані; Програма пропонує користувачу обрати файл; Програма зчитує дані з обраного файлу.
Альтернативні потоки	Помилка при відкритті файлу. Якщо виникає помилка при відкритті файлу, програма відображає повідомлення про помилку, та користувач має можливість змінити вибір. Помилка при отриманні даних. Якщо виникла помилка при отриманні даних (невідповідність формату), програма відображає повідомлення про помилку отримання даних.
Постумови	Передає отримані дані для подальшого розрахунку таким варіантам використання, як «Розрахувати характеристики вибірки», «Розрахувати регресію», «Переглянути точки вихідної вибірки у вигляді списку чисел» та «Переглянути розраховану регресію у вигляді списку чисел».

Далі переходимо до розробки сценаріїв основних варіантів використання програми для оцінювання метрики LOC програмних застосунків на C++.

2.1.3 Сценарії основних варіантів використання

Для моделювання процесу виконання операцій використовуються діаграми діяльності. Діаграми діяльності для основних варіантів використання програми для оцінювання метрики LOC програмних застосунків на C++ представлені на рис. 2.2 – 2.3. В якості основних варіантів використання представлені варіанти використання «Розрахувати регресію» та «Розрахувати значення метрики LOC»

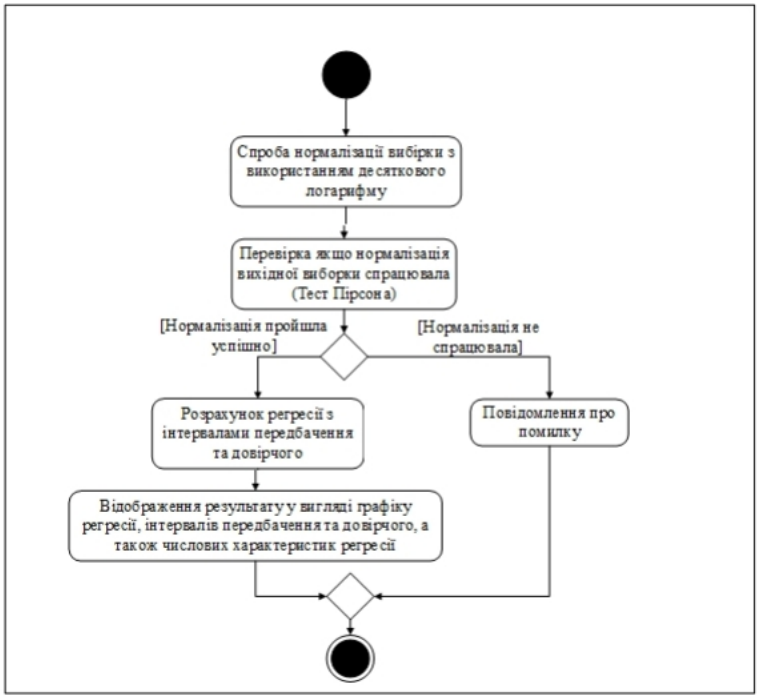


Рисунок 2.2 – Діаграма діяльності для варіанту використання «Розрахувати регресію»

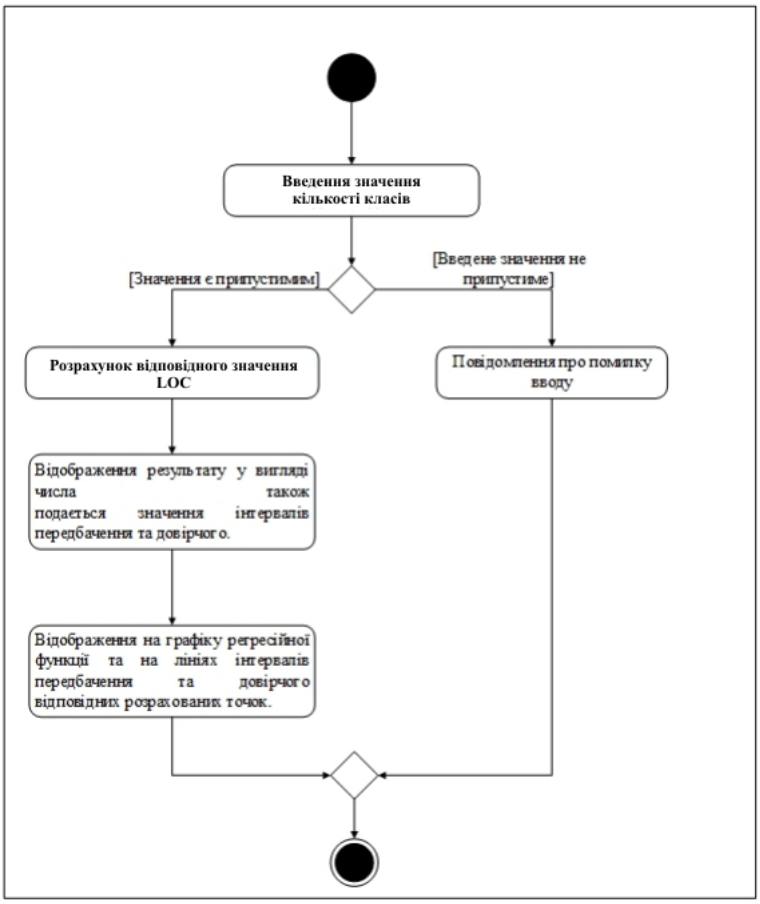


Рисунок 2.3 – Діаграма діяльності для варіанту використання «Розрахувати значення метрики LOC»

Розробку основних варіантів використання програми для оцінювання метрики LOC програмних застосунків на C++завершено.

2.2 Архітектура програми для оцінювання метрики LOC програмних застосунків на C++

Оскільки програма для оцінювання метрики LOC програмних застосунків на C++ буде містити інтерфейс користувача (User Interface - UI), реалізований з використанням WinForms, то в якості архітектури такої програми найкраще підійде шаблон MVC. У такому разі:

- компонент Model представляє дані та логіку (Regression, ProjectsTable);
- компонент View реалізує віконні форми (UI);
- компонент Controller керує логікою між UI та моделлю.

Архітектура програми для оцінювання метрики LOC програмних застосунків на C++ представлена на рис. 2.4.

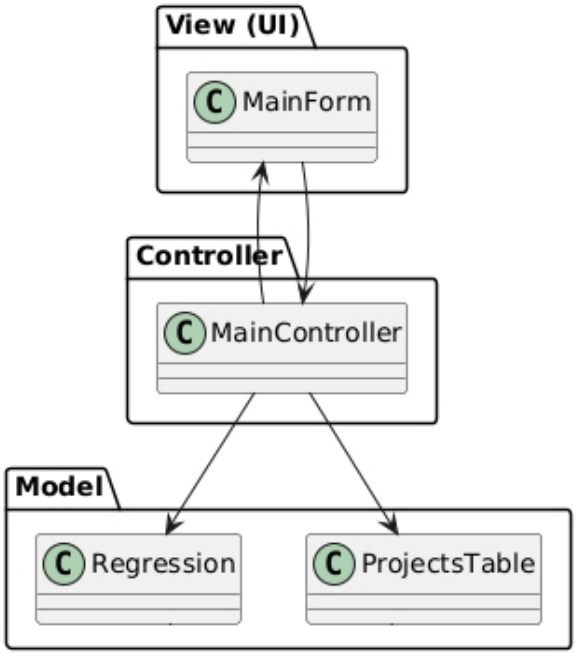


Рисунок 2.4 – Архітектура програми для оцінювання метрики LOC програмних застосунків на C++

Опишемо розподіл відповідальності між шарами шаблону MVC програми для оцінювання метрики LOC програмних застосунків на C++.

Шар Model:

Regression.cs - логіка побудови регресійної моделі та прогнозування;

ProjectsTable.cs - завантаження даних та сутність з NumClasses і LOC.

Шар View:

MainForm.cs (WinForms);

Елементи UI, які містять в тому числі, кнопки "Histograms", "Regression", "Output", "Calculate", поле для вводу кількості класів та ін.

Controller:

MainController.cs - реагує на події з View, викликає Model для обчислень, оновлює View результатами.

2.3 Розробка проєкту програми для оцінювання метрики LOC програмних застосунків на C++

2.3.1 Ескізний проєкт програми

Оскільки сценарії основних варіантів використання програми для оцінювання метрики LOC програмних застосунків на C++ вже були представлені раніше у підрозділі 2.1.3 Сценарії основних варіантів використання, в рамках ескізного проєкту програми розробимо прототип користувальницького інтерфейсу.

Прототип користувальницького інтерфейсу

Графічний інтерфейс, який буде мати розроблювана програма для оцінювання метрики LOC програмних застосунків на C++, має дуже важливе значення. Він повинен бути простим та водночас зручним для використання, щоб будь-який користувач, який має базові навички роботи з комп’ютером, міг би без

жодних перешкод працювати з програмою.

Виконаємо проєктування користувацького графічного інтерфейсу (UI).

Спочатку виконаємо проєктування головного вікна програми, яке наведено на рисунку 2.5.

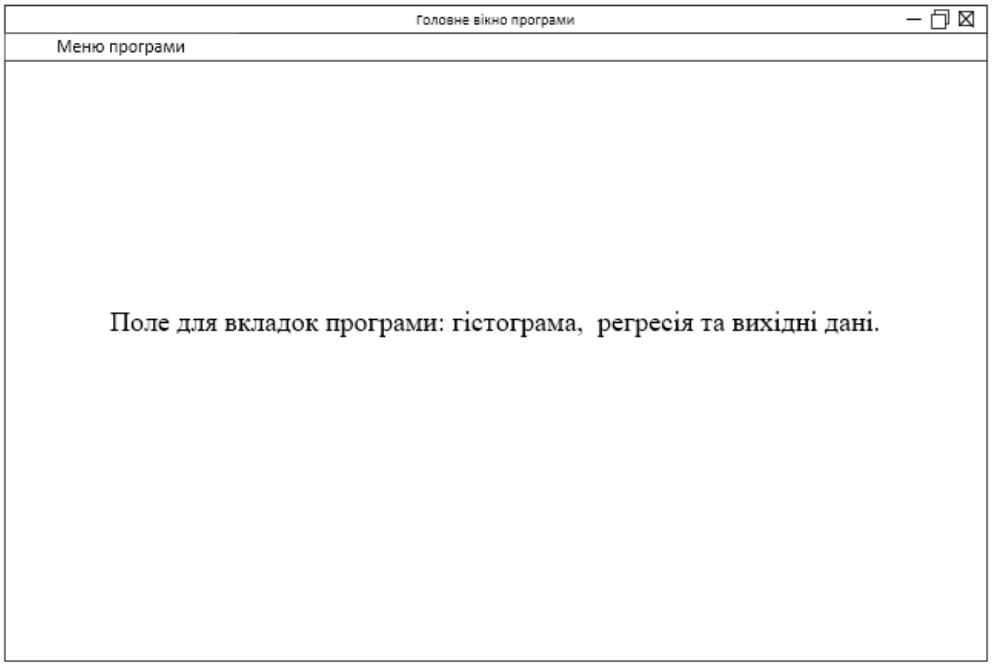


Рисунок 2.5 – Проєкт головного вікна програми для оцінювання метрики LOC програмних застосунків на C++

Наступним кроком виконуємо проєктування вкладок: гістограма, регресія, вихідні дані та розрахунок значення метрики LOC. Проєкт вкладок "Гістограма" та "Вихідні дані", яке наведено на рисунках 2.6 та 2.7 відповідно.

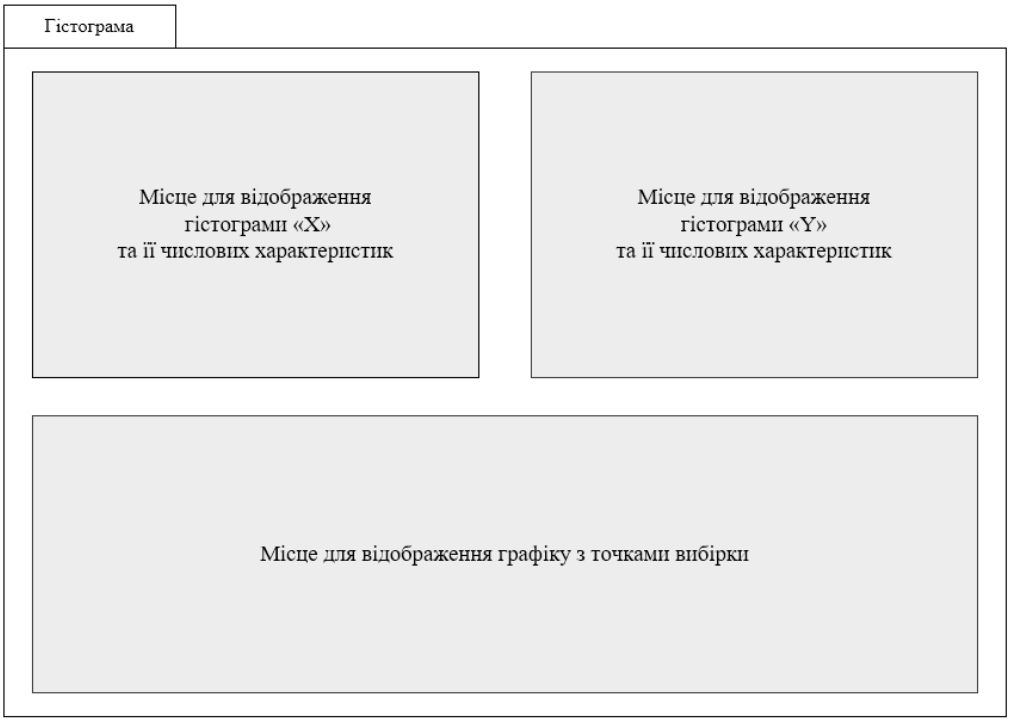


Рисунок 2.6 – Проєкт вкладки «Гістограма» програми для оцінювання метрики LOC програмних застосунків на C++

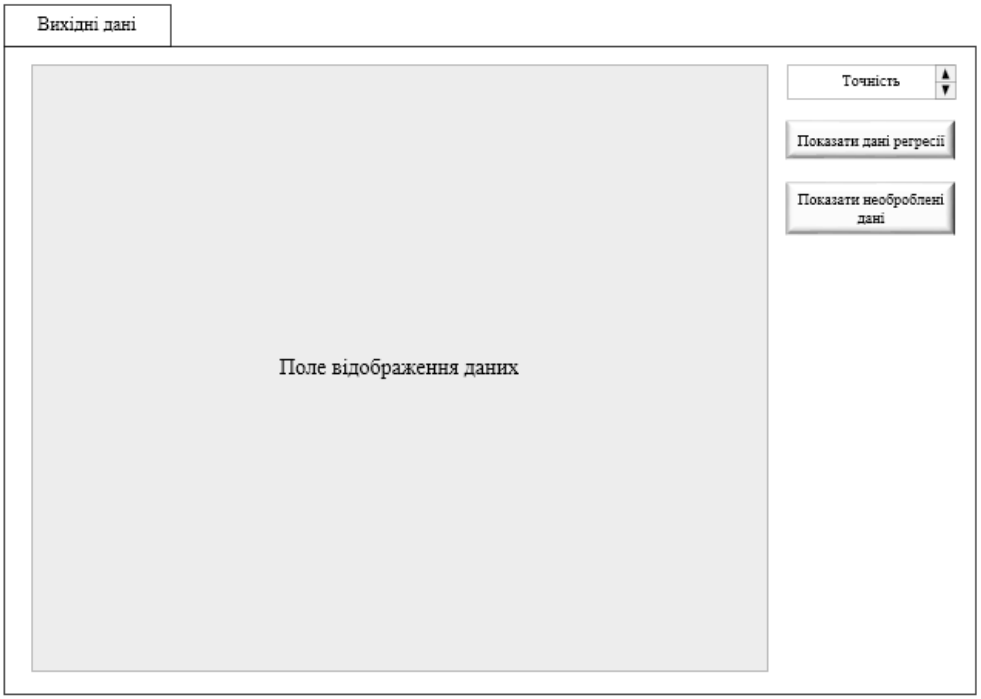


Рисунок 2.7 – Проєкт вкладки «Вихідні дані» програми для оцінювання метрики LOC програмних застосунків на C++

Після виконання ескізного проєкту можна переходити до розробки технічного проєкту програми для оцінювання метрики LOC програмних застосунків на C++.

2.3.2 Технічний проєкт програми

Статична модель програми

Технічний проєкт програми для оцінювання метрики LOC програмних застосунків на C++ розпочнемо з розробки статичної моделі програмного забезпечення.

Статична модель програми для оцінювання метрики LOC програмних застосунків на C++ представлена діаграмою класів, яка наведена на рис. 2.8.

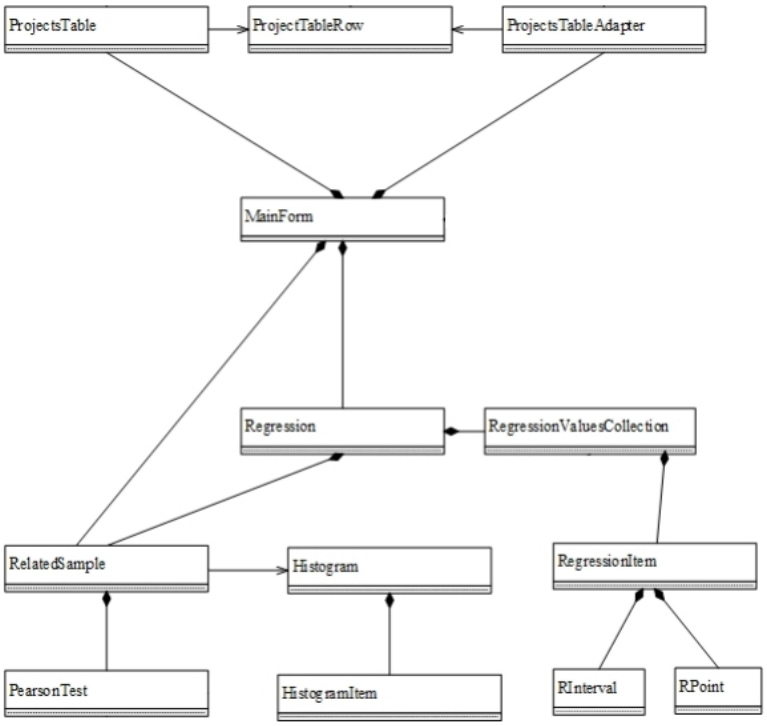


Рисунок 2.8 – Діаграма класів програми для оцінювання метрики LOC програмних застосунків на C++

Динамічна модель програми

Наступним кроком розробимо динамічну модель програми для оцінювання метрики LOC програмних застосунків на C++. Динамічна модель програми

представлена діаграмами послідовності. Діаграми послідовності для варіантів використання «Розрахувати регресію» та «Розрахувати значення метрики LOC» наведені на рисунках 2.9 та 2.10 відповідно.

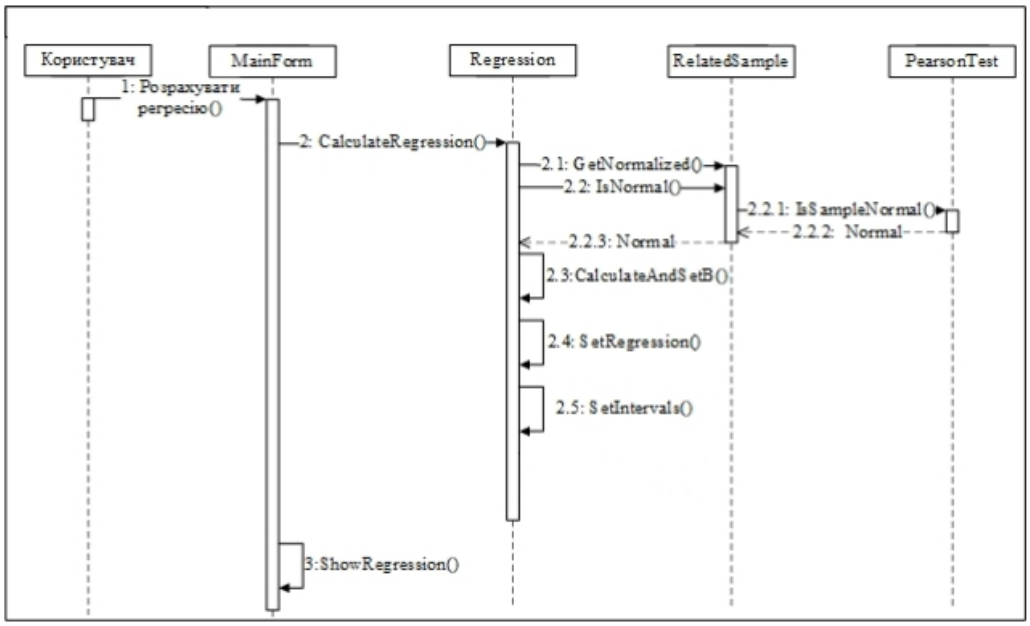


Рисунок 2.9 – Діаграма послідовності для варіанту використання «Розрахувати регресію»

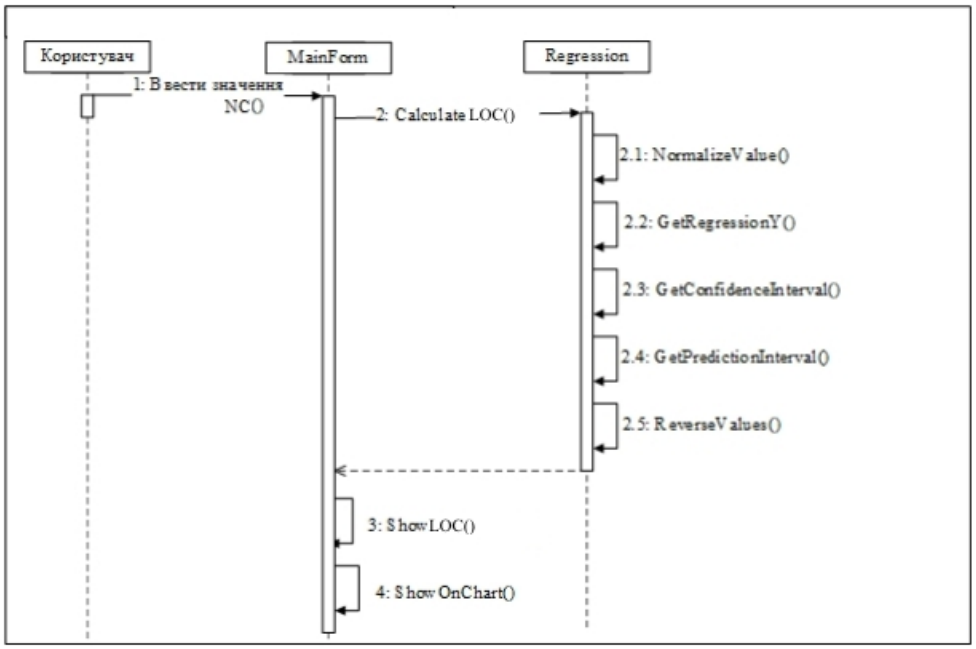


Рисунок 2.10 – Діаграма послідовності для варіанту використання «Розрахувати значення метрики LOC»

На цьому розробку технічного проєкту програми для оцінювання метрики LOC програмних застосунків на C++ завершено, можна переходити до розробки робочого проєкту.

2.3.3 Робочий проєкт програми

Вибір програмних засобів розробки

Робочий проєкт програми для оцінювання метрики LOC програмних застосунків на C++ розпочнемо з вибору програмних засобів розробки.

Для написання програми була обрана мова програмування C# [7] та програмна платформа .NET Framework [8]. Ця платформа завдяки великій кількісті компонентів та класів дозволяє в стислий термін розробити складну систему.

Завдяки тому, що програма розроблялась виключно для операційної системи Windows та призначається для виконання на комп`ютері (настільному чи ноутбуку), для створення графічного інтерфейсу цілком підходить технологія WindowsForms [9]. WindowsForms – це бібліотека класів, відповідальна за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Дана технологія спрощує доступ до елементів інтерфейсу Microsoft Windows за допомогою створення обгортки для Win32 API в керованому коді. Це дуже спрощує розробку та скорочує її термін завдяки зручним класам, з якими можна працювати, повністю абстрагуючись від взаємодією с операційною системою та викликів системних функцій.

Для відображення графіків, діаграм та гістограм був обраний програмний компонент LiveCharts [10], який розповсюджується під ліцензією MIT [11], та є вільним для використання у будь-яких проєктах. Цей компонент призначено для використання в програмах, які виконуються на платформі .NET Framework. LiveCharts надає можливість відображати гістограми, діаграми точок та графіки.

Ці види діаграм потрібні у даному проєкті для відображення гістограм розподілу випадкової величини, графіку регресії та відображення точок вибірки.

В якості середовища для розробки програми було обрано Microsoft Visual Studio [12]. Це інтегроване середовище розробки програмного забезпечення та низка інших інструментальних засобів, таких, як редактор вихідного коду, вбудований налагоджувач, компілятор, засоби для роботи з базою даних, підключення плагінів, управління файлами проєкту, система контроля версій та інше. Visual Studio дозволяє зручно поєднати всі компоненти, необхідні для розробки даної програми.

Реалізація та тестування основних класів програми

Наступним кроком розробки програми для оцінювання метрики LOC програмних застосунків на C++ виконаємо реалізацію та тестування основних класів програми.

Програма розроблена мовою програмування C#. Код програми наведено у додатку Б – Код програми для оцінювання метрики LOC програмних застосунків на C++.

Для тестування функцій програми було обрано техніку еквівалентної розбивки. Нижче представлено тестування функції «Розрахувати значення метрики LOC».

Були виділені класи еквівалентності, які представлені в таблиці 2.7. Результати тестування правильних класів еквівалентності та неправильних класів еквівалентності представлені в таблицях 2.8 та 2.9 відповідно.

Таблиця 2.7 – Класи еквівалентності для функції «Розрахувати значення метрики LOC»

Вхідні данні	Правильні класи	Неправильні класи
Кількість класів	Введено числове значення в межах допустимих значень (1)	Значення не введено (2), введено значення поза межами допустимих значень (3)

Таблиця 2.8 – Тести правильних класів еквівалентності для функції «Розрахувати значення метрики LOC»

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	1	Кількість класів : 10	Отримано розрахунок значення метрики LOC.	Тест пройдено

Таблиця 2.9 – Тести неправильних класів еквівалентності для функції «Розрахувати значення метрики LOC»

№ тесту	Покритий клас	Вхідні дані	Вихідні данні	Результат тестування
1	2	Кількість класів : – (поле не заповнено)	Повідомлення про те, що поле повинно бути заповнено	Тест пройдено
2	3	Кількість класів: -1	Повідомлення про те, що введено недопустиме значення	Тест пройдено

Випробування програмного забезпечення

Завершивши етап тестування програми для оцінювання метрики LOC програмних застосунків на C++, проводимо випробування програми.

Відповідно до програми та методики випробування, яка представлена у додатку Д, було проведено випробування програми для оцінювання метрики LOC програмних застосунків на C++. Фрагмент випробування деяких функцій програми наведено нижче.

Перевірка дії «Нормалізувати дані десятковим логарифмом»

У вкладці «Histograms» – у пункті головного меню «Sample» обрали пункт «Logarithm 10».

Результатом цієї дії стало вивід на вкладці «Histograms» наступних даних:

- гістограма нормалізованих даних кількості класів (X);
- числові характеристики нормалізованих даних кількості класів;
- гістограма нормалізованих даних значення метрики LOC (Y);
- числові характеристики нормалізованих даних значення метрики LOC;
- точки нормалізованих даних вибірки.

Перевірка дії «Розрахувати значення метрики LOC»

Для перевірки розрахунку значення метрики LOC на вкладці «Regression», у блоці «Calculation» в полі «NC» ввели значення «20», та натиснули кнопку «Calculate».

Результатом цієї дії стало вивід значення метрики LOC, довірчого інтервалу та інтервалу прогнозування у блоці «Calculation», що наведено на рисунку 2.11.

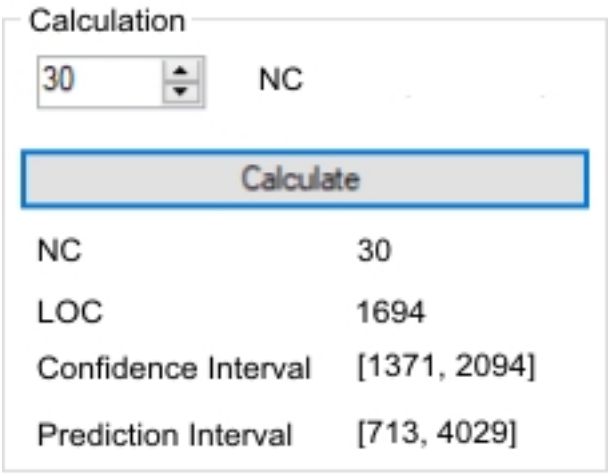


Рисунок 2.11 – Блок «Calculation» з розрахунками

На цьому розробку програми для оцінювання метрики LOC програмних застосунків на C++ можна вважати завершеною.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ МЕТРИКИ LOC ПРОГРАМНИХ ЗАСТОСУНКІВ НА C++

В представленій кваліфікаційній роботі була розв’язана практична задача інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розроблена програма для оцінювання метрики LOC програмних застосунків на C++.

В результаті розробки була створена програми для оцінювання метрики LOC програмних застосунків на C++. Назва програми: «CalcLOC». Інтерфейс програми було розроблено англійською мовою. Проведено тестування та випробування програми.

Блок «Calculation» з розрахунком значення метрики LOC та границь довірчого інтервалу та інтервалу прогнозування наведено на рисунку 3.1.

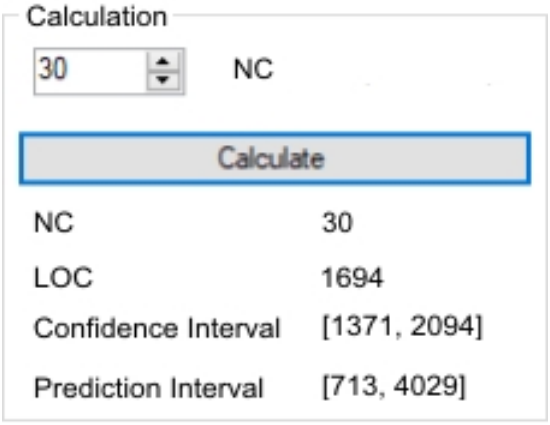


Рисунок 3.1 – Блок «Calculation» з розрахунком значення метрики LOC та границь довірчого інтервалу та інтервалу прогнозування

Програма має експлуатуватись на обладнанні з параметрами не нижчими за наступні:

– CPU: Intel(R) i5 2,16 МГц або аналогічний за параметрами AMD;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

Програма «CalcLOC» створена для прискорення розрахунків метрики LOC програмних застосунків на C++. Призначення програми полягає можливості користувачам швидко отримувати прогнозне значення метрики LOC C++-проєктів, що, в свою чергу, може бути використано для оцінки продуктивності команд розробників, технічного боргу, а також при порівнянні різних версій одного й того самого програмного забезпечення.

Програма надає можливість виконувати наступні функції:

розрахувати характеристики вихідної вибірки;

побудувати гістограми вихідної вибірки та графік з її точками;

розрахувати регресію, довірчий інтервал та інтервал прогнозування;

побудувати графік регресії, довірчого інтервалу та інтервалу прогнозування;

розрахувати значення метрики LOC за введеним значенням кількості класів;

переглянути точки отриманої вибірки, представлені числовими значеннями;

переглянути розраховані точки регресії, довірчого інтервалу та інтервалу прогнозування у вигляді числових значень.

Для запуску програмного забезпечення «CalcLOC» необхідно запустити файл з назвою CalcLOC.exe.

Була розроблена наступна програмна документація на програму для оцінювання метрики LOC програмних застосунків на C++: технічне завдання, код програми, опис програми, інструкція користувача, програма та методика випробування програми, що наведені у додатках А, Б, В, Г, Д відповідно.

4 ОХОРОНА ПРАЦІ

4.1 Вступ

Охорона праці - це комплексна система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, метою якої є забезпечення збереження життя, здоров’я та працездатності працівника під час виконання ним трудових обов’язків.

Вона охоплює всі сфери виробничої діяльності та спрямована на створення безпечних і здорових умов праці, попередження нещасних випадків, професійних захворювань, зменшення виробничих ризиків і поліпшення загального добробуту працівників.

Ключовим нормативно-правовим актом у цій галузі є Закон України «Про охорону праці» [13]. Цей закон регламентує основні засади реалізації конституційного права кожного працівника на захист його життя і здоров’я в процесі трудової діяльності. Він також встановлює обов’язки та відповідальність роботодавця за створення безпечних умов праці, визначає механізми державного контролю, регулює участь органів державної влади у формуванні політики безпеки праці та забезпечує єдиний підхід до організації системи охорони праці в Україні.

Крім того, закон сприяє формуванню культури безпеки на робочому місці та зобов’язує всі суб’єкти господарювання дотримуватись встановлених норм і стандартів, забезпечувати працівників засобами індивідуального захисту, проводити інструктажі та навчання з питань охорони праці.

На підприємстві управління охороною праці реалізується через системний підхід до організації безпечних умов праці, дотримання вимог законодавства, впровадження профілактичних та інженерно-технічних заходів. Особлива увага приділяється створенню сприятливих умов при роботі з комп’ютерною технікою:

оптимальний мікроклімат, достатнє освітлення, мінімізація шуму, захист від шкідливих випромінювань та дотримання вимог електробезпеки.

4.2 Структура управління охороною праці на підприємстві

Управління охороною праці - це процес організації, планування та впровадження комплексу заходів правового, організаційного, технічного, санітарно-гігієнічного й профілактичного характеру, спрямованих на забезпечення здорових і безпечних умов праці, збереження життя та працездатності працівників.

Згідно зі статтею 13 Закону України "Про охорону праці", роботодавець зобов’язаний створити на кожному робочому місці умови праці, що відповідають нормативно-правовим актам, а також забезпечити дотримання прав працівників у сфері охорони праці. У випадку, якщо на підприємстві працює понад 50 осіб, обов’язковим є створення служби охорони праці.

Для реалізації ефективного управління охороною праці на підприємстві роботодавець повинен забезпечити функціонування відповідної системи, яка передбачає:

- формування служб з охорони праці та призначення відповідальних осіб, з визначенням їхніх обов’язків і відповідальності;

- розроблення та реалізацію комплексних заходів для досягнення нормативів безпеки праці й підвищення рівня захищеності персоналу;

- адаптацію профілактичних заходів відповідно до змін умов виробництва;

- впровадження сучасних технологій, механізації, автоматизації, досягнень науки й техніки, а також принципів ергономіки;

- підтримання технічного стану будівель, споруд, обладнання та забезпечення регулярного моніторингу їх безпечності;

- аналіз причин нещасних випадків та захворювань, реалізацію заходів із їх попередження;

- організацію аудитів з охорони праці, санітарно-гігієнічних досліджень, технічної експертизи та атестацій робочих місць;

- розробку та затвердження локальних нормативних актів (інструкцій, положень), що регламентують безпечну поведінку працівників;

- здійснення контролю за дотриманням технологічних процесів, правил експлуатації обладнання, використання засобів індивідуального та колективного захисту;

- проведення навчання, інструктажів, інформаційної роботи з питань безпечного виконання робіт;

- організацію негайної допомоги потерпілим у разі аварій, залучення аварійно-рятувальних служб.

Роботодавець несе особисту відповідальність за недотримання вимог охорони праці.

4.3 Забезпечення техніки безпеки та охорона праці оператора персонального комп`ютера

Для забезпечення техніки безпеки слід дотримуватись вимог, які описані нижче.

Загальні положення

Для безпечної роботи з персональним комп’ютером працівник повинен:

- мати базові навички роботи з персональним комп`ютером та знати інструкцію з його експлуатації;

- пройти вступний і первинний інструктаж з охорони праці;

- не мати медичних протипоказань до роботи з персональним комп`ютером;

- знати можливі шкідливі фактори: електромагнітне випромінювання, стомлення зору, статичне навантаження;

- дотримуватись правил електробезпеки, інструкцій підприємства, техніки

пожежної безпеки;

- уміти надавати першу допомогу при ураженні електричним струмом, знати розташування засобів пожежогасіння.

Організація робочого місця

Відстань між моніторами по тилу - не менше 2 м; між боками - від 1,2 м.

Екран розташовують так, щоб уникати відблисків, бажано — не напроти вікна.

Висота верхнього краю екрана - на рівні очей або трохи нижче.

Відстань до екрана - 60–70 см, мінімально - 50 см.

Клавіатура повинна розміщуватись так, щоб руки залишались у природному положенні. За потреби - використання підставок.

Робоче крісло повинно бути регульованим по висоті, куту нахилу сидіння й спинки.

Кабелі слід розміщувати так, щоб уникнути їх пошкодження.

Для запобігання накопиченню статичної електрики - використовуються нейтралізатори, зволожувачі повітря, кімнатні рослини, акваріуми.

Вологість у приміщенні - у межах 40–60%; обов’язкова вентиляція.

Режим праці та відпочинку

Безперервна робота з персональним комп`ютером не повинна перевищувати 2 годин.

У нічну зміну (з 22:00 до 6:00) тривалість перерв збільшується на 60 хв.

У перервах рекомендується виконувати вправи для зниження втоми й напруги.

За високого навантаження рекомендована психологічна розрядка у спеціальних кімнатах.

Вимоги безпеки перед початком роботи

Перед початком зміни користувач персонального комп`ютера зобов’язаний:

- підготувати робоче місце, перевірити освітлення;

- оглянути обладнання на наявність пошкоджень кабелів, кнопок, корпусу;

- у разі виявлення несправностей - повідомити керівника та не приступати до роботи до їх усунення.

Вимоги під час роботи

У процесі роботи з комп’ютером слід:

- суворо дотримуватися інструкцій з експлуатації;

- не залишати персональний комп`ютер увімкненим без нагляду;

- при тривалих перервах - вимикати персональний комп`ютер з мережі;

- у разі несправностей - зупинити роботу, вимкнути персональний комп`ютер, повідомити технічний персонал;

- за можливості - чергувати роботу за комп’ютером з іншими видами діяльності;

- для роботи з документами використовувати підставки, встановлені на рівні екрана;

- під час перерв - виконувати рекомендовані вправи для очей і спини.

Дії в аварійних ситуаціях

У разі виникнення аварійної ситуації:

- Загальна аварія: негайно зупинити роботу, знеструмити обладнання, повідомити керівника, вжити заходів для ліквідації загрози.

- Травмування працівника: усунути небезпечний фактор (електрика, вогонь), надати першу допомогу, викликати медиків, повідомити керівництво.

Пожежа:

- знеструмити обладнання (персональний комп`ютер, розетки);

- врятувати документи й важливі речі;

- допомогти постраждалим;

- повідомити керівництво;

- за потреби - викликати пожежну службу (101);

- долучитись до гасіння пожежі.

Коротке замикання або падіння обладнання:

- вимкнути ПК, витягти штекер з розетки;

- повідомити відповідального фахівця;

- не включати обладнання до приїзду спеціаліста.

Протікання води (з опалення або водопроводу):

- вимкнути електроживлення, комп'ютери, освітлення;

- перемістити техніку у безпечне місце;

- повідомити керівника та відповідальні служби.

Землетрус:

- вимкнути персональний комп`ютер, заховати документи, залишити приміщення по сходах (ліфтом користуватися заборонено).

Ураження електрострумом:

- негайно вимкнути живлення;

- якщо потерпілий не дихає - виконувати штучне дихання та непрямий масаж серця;

- викликати швидку допомогу.

Дії після завершення роботи

Вимкнути персональний комп`ютер та інші електроприлади (принтер, обігрівач, кондиціонер, шредер тощо).

Закрити документи й папери в сейф.

Провести прибирання робочого місця.

Закрити вікна, двері, кватирки.

Вимкнути освітлення.

Здати приміщення під охорону.

ВИСНОВКИ

Метою кваліфікаційної роботи була розробка програми для оцінювання метрики LOC програмних застосунків на C++, яка передбачає розв’язання практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов.

У ході виконання кваліфікаційної роботи було зроблено аналіз практичної задачі інженерії програмного забезпечення, що стосується теми роботи, проаналізовано існуючі аналоги та обґрунтовано необхідність розробки програми, здійснено постановку задачі на розробку програми для оцінювання метрики LOC програмних застосунків на C++, на основі постановки задачі розроблено технічне завдання на розробку програми.

Були розроблені архітектура програми, ескізний, технічний та робочій проєкти програми для оцінювання метрики LOC програмних застосунків на C++. Здійснено кодування, тестування та випробування розробленої програми. Також розроблена необхідна документація до програми.

Представлені результати розробки програми для оцінювання метрики LOC програмних застосунків на C++.

Також був виконаний розділ з охорони праці.

Розроблена програма дозволить користувачам швидко отримувати прогнозне значення метрики LOC C++-проєктів.

Мета кваліфікаційної роботи досягнута, усі завдання виконані в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 1061-1998-IEEE Standard for a Software Quality Metrics Methodology. URL: https://ieeexplore.ieee.org/document/749159 (дата звернення: 20.03.2025).

2. TIOBE Index for December 2024. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 20.03.2025).

3. Math. Graphics. Programming. URL: https://ch.mathworks.com/products/matlab.html (дата звернення: 20.03.2025).

4. Завантаження надбудови "Пакет аналізу" в Excel. URL: https://support.microsoft.com/uk-ua/office завантаження-надбудови-пакет-аналізу-в-excel-6a63e598-cd6d-42e3-9317-6b40ba1a66b4 (дата звернення: 20.03.2025).

5. Maximize productivity by fixing the tech debt that truly slows you down. URL: https://codescene.com/ (дата звернення: 20.03.2025).

6. Yan Xin, Su Xiao Gang. Linear regression analysis: theory and computing. Singapore: World Scientific Publishing Co. Pte. Ltd., 2009. 328 p.

7. C# documentation. URL: https://docs.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 20.04.2025).

8. .NET documentation. URL: https://docs.microsoft.com/en-us/dotnet/ (дата звернення: 20.04.2025).

9. Windows Forms. URL: https://docs.microsoft.com/en-us/dotnet/framework/winforms/ (дата звернення: 20.04.2025).

10. LiveCharts. URL: https://lvcharts.net/ (дата звернення: 20.04.2025).

11. The MIT License. URL: https://github.com/Live-Charts/Live-Charts/blob/master/LICENSE.TXT (дата звернення: 20.04.2025).

12. Visual Studio. URL: https://visualstudio.microsoft.com/ (дата звернення: 20.04.2025).

13. Про охорону праці : Закон України від 21.11.2002 р. № 229-IV. Дата оновлення: 04.02.2021. URL: https://zakon.rada.gov.ua/laws/main/2694-12#Text. (дата звернення: 20.03.2025).

Додаток А

Технічне завдання на розробку програми для оцінювання метрики LOC програмних застосунків на C++

Вступ

Назва розроблюваного проєкту: «Програма для оцінювання метрики LOC програмних застосунків на C++». Область застосування – підприємство, яке займається розробкою відповідного програмного забезпечення.

1. Підстави для розробки

Підставою для розробки програмного забезпечення є наказ на затвердження тем кваліфікаційних робіт бакалаврів зі спеціальності 121 Інженерія програмного забезпечення у Національному університеті кораблебудування імені адмірала Макарова.

2. Призначення розробки

2.1 Експлуатаційне призначення

Експлуатаційним призначенням програми є спрощення розрахунку оцінки метрики LOC програмних застосунків на C++ і візуалізація результатів розрахунку.

2.2 Функціональне призначення програми

Функціональним призначенням програми є автоматизація процесів:

– розрахунку числових характеристик вибірки, таких, як: кількість значень, вибіркове середнє, дисперсія, асиметрія, ексцес, результат тесту Пірсона на нормальність розподілу випадкової величини;

– побудови гістограми вихідної вибірки та графіку з її точками;

– розрахунку числових характеристики регресії, таких, як: коефіцієнти рівняння регресії (b_0 та b_1), коефіцієнт детермінації (R_2), середнє значення

відносної похибки (*MMRE*), відсоток значень, які мають відносну похибку 0,25 (*PRED*(0,25));

– розрахунок значень границь довірчого інтервалу та інтервалу прогнозування;

– побудова графіку регресії, довірчого інтервалу та інтервалу прогнозування;

– розрахунок значення метрики LOC за введеним значенням кількості класів;

– перегляд точок отриманої вибірки, представлених числовими значеннями;

– перегляд розрахованих точок регресії, довірчого інтервалу та інтервалу прогнозування у вигляді числових значень.

3. Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинно виконувати наступні функції:

розрахувати числові характеристики вибірки, такі, як: кількість значень, вибіркове середнє, дисперсія, асиметрія, ексцес, результат тесту Пірсона на нормальність розподілу випадкової величини;

побудувати гістограми вихідної вибірки та графік з її точками;

– розрахувати числові характеристики регресії, такі, як: коефіцієнти рівняння регресії (b_0 та b_1), коефіцієнт детермінації (R_2), середнє значення відносної похибки (*MMRE*), відсоток значень, які мають відносну похибку 0,25 (*PRED*(0,25));

розрахувати значення границь довірчого інтервалу та інтервалу прогнозування;

побудувати графік регресії, довірчого інтервалу та інтервалу прогнозування;

розрахувати значення метрики LOC за введеним значенням кількості класів;

переглянути точки отриманої вибірки, представлені числовими значеннями;

переглянути розраховані точки регресії, довірчого інтервалу та інтервалу прогнозування у вигляді числових значень.

3.1.2 Вимоги до організаційних вхідних та вихідних даних

Для розрахунку регресії введення даних здійснюється із файлу формату .csv, який містить наступні стовпці: значення кількості класів NC, значення метрики LOC.

Для прогнозування значення метрики LOC згідно регресії введення даних здійснюється до відповідного поля вводу вручну з клавіатури.

Вхідні дані:

для розрахунку регресії: кількість класів, LOC;

для прогнозування значення метрики LOC згідно регресії: кількість класів.

Вихідні дані:

числові характеристики вибірки, такі, як: кількість значень, вибіркове середнє, дисперсія, асиметрія, ексцес, результат тесту Пірсона на нормальність розподілу випадкової величини;

гістограми вибірки;

відповідні точки вибірки;

числові характеристики регресії, такі, як: коефіцієнти рівняння регресії (b_0 та b_1), коефіцієнт детермінації (R_2), середнє значення відносної похибки (*MMRE*), відсоток значень, які мають відносну похибку 0,25 (*PRED*(0,25));

графік, який відображає лінію регресійної функції, довірчий інтервал, інтервал прогнозування, точки даних;

значення метрики LOC;

відображення на графіку моделі розрахованої точки.

3.2 Вимоги до надійності

Програмне забезпечення повинно функціонувати при безперебійній роботі персонального комп’ютера. При виникненні збоїв в роботі відновлення нормальної роботи повинне проводитися після перезавантаження програмного забезпечення.

У разі введення користувачем некоректної інформації програмне забезпечення повинно повідомити про помилку і надати можливість виправити її.

3.3 Вимоги до умов експлуатації

Необхідний рівень підготовки користувача: мінімальні навики в користуванні комп’ютером. Комп’ютер призначений для роботи в закритому опалювальному приміщені.

3.4 Вимоги до складу та параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплекції:

– CPU: Intel(R) i5 2,16 МГц або аналогічний за параметрами AMD;

– RAM: 4 GB;

– HDD: 10 GB вільного місця.

3.5 Вимоги до інформаційної та програмної сумісності

Для повноцінного функціонування програми необхідно використовувати ОС Windows версії не нижче 7, .Net Framework 4.8.1.

3.6 Вимоги до маркування та пакування

Додаток буде упаковано в електронний архів і мати найменування CalcLOC.zip.

3.7 Вимоги до транспортування та зберігання

Вимоги до транспортування не висовуються у зв’язку з відсутністю фізичних носів.

4. Вимоги до програмної документації

Програмна документація має містити наступні документи:

- технічне завдання;
- опис програми;
- код програми;
- інструкція користувача;
- програма і методика випробування програми.

5. Техніко-економічні показники

Не розраховуються в зв’язку з тим, що продукт розробляється в рамках кваліфікаційної роботи.

6. Стадії та етапи розробки

Стадії та етапи розробки представлені нижче в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки проєкту

№ п/п	Назва етапів роботи	Кінцевий термін виконання
1.	Аналіз практичної задачі інженерії ПЗ	17.02.2025
2.	Аналіз вимог до ПЗ	28.03.2025
3.	Розробка архітектури ПЗ	08.04.2025
4.	Розробка проєкту ПЗ	15.04.2025
5.	Реалізація та тестування ПЗ	28.04.2025

7. Порядок контролю та прийому

Контроль за аналізом та проєктуванням кожної окремої частини програмного забезпечення на кожному етапі з урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені

строки та узгоджена із замовником.

Прийом проводяться відповідно до програми і методики випробувань і документують за допомогою протоколу проведення випробувань. У разі знаходження помилок підчас прийому програмного виробу складається акт про знайдені помилки, який підписуються представниками замовника і розробника і затверджуються керівником організації – замовника та організації – розробника. Розробник повинен на протязі не більше ніж 2 тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

Додаток Б

Код програми для оцінювання метрики LOC програмних застосунків на C++

Вміст файлу вихідного коду «Regression.cs»:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms.DataVisualization.Charting;
namespace CalcLOC
{
    class Regression
    {
        private readonly RelatedSample[] _sampleArray;
        public decimal B0 { get; private set; }
        public decimal B1 { get; private set; }

        private decimal S { get; set; }
        private decimal TDistribution { get; set; }

        private decimal MeanNormalizedX { get; set; }

        private decimal SumSquaredDiffXAndMean { get; set; }

        public RelatedSample RawSample
        {
            get => _sampleArray[0];
            set => _sampleArray[0] = value;
        }

        public RelatedSample RegressionSample
        {
            get => _sampleArray[1];
            set => _sampleArray[1] = value;
        }

        public RegressionValuesCollection RegressionValues { get;  }
```

```
        public decimal Alpha { get; }

        private readonly Func<double, double> _logarithm;
        private readonly Func<double,double> _inverseFunc;

        public bool HasRegressionBeenCalculated { get; private set; }

        public Regression(RelatedSample sample, Func<double, double> logarithm,
Func<double, double> inverseFunc,
            decimal alpha = 0.05M)
        {
            _logarithm = logarithm;
            _inverseFunc = inverseFunc;
            _sampleArray = new RelatedSample[2];
            Alpha = alpha;
            RegressionValues = new RegressionValuesCollection();
            RawSample = (RelatedSample) sample.Clone();
            RawSample.SortByX();
            HasRegressionBeenCalculated = false;
        }

        private RelatedSample GetNormalizedSample()
        {
            var result = RawSample.GetNormalized(_logarithm);
            if(!result.X.IsNormal()) throw new ArgumentException("Can't normalize
the sample");
            return result;
        }



        private void ReverseConversion(RelatedSample sample)
        {
            sample.X.Clear();
            sample.X.AddRange(RawSample.X.GetCopyRandomValues());
            sample.Y.Normalize(_inverseFunc);
        }

        private double ReverseValue(decimal val)
        {
            return  _inverseFunc((double)val);
```

```
        }

        private  void CalculateAndSetB(RelatedSample normalizedSample)
        {
            decimal sumZxiZyi = 0;
            decimal sumZxi = 0;
            decimal sumZyi = 0;
            decimal sumZxi2 = 0;
            decimal n = normalizedSample.Count;
            for (int i = 0; i < n; i++)
            {
                sumZxi += normalizedSample.X[i];
                sumZyi += normalizedSample.Y[i];
                sumZxiZyi += normalizedSample.X[i] * normalizedSample.Y[i];
                sumZxi2 += normalizedSample.X[i] * normalizedSample.X[i];
            }

            B1 =  (n * sumZxiZyi - sumZxi * sumZyi) / (n * sumZxi2 - sumZxi *
sumZxi);
            B0 = (sumZyi - B1 * sumZxi) / n;
        }

        public double NonLinear(double x)
        {
            return _inverseFunc((double)B0) * Math.Pow(x, (double)B1);
        }

        public decimal Linear(decimal x)
        {
            return B1 * x + B0;
        }
        private decimal GetS(RelatedSample normalizedRegression,RelatedSample
normalizedSample)
        {
            decimal sum = 0;
            for (int i = 0; i < normalizedRegression.Count; i++)
            {
                var diff = normalizedSample.Y[i] - normalizedRegression.Y[i];
                sum += diff * diff;
            }
```

```
            return (decimal)Math.Sqrt((double) (1M / (normalizedRegression.Count -
2) * sum));
        }


        private decimal GetInterval(decimal x, decimal s, decimal t, int n,
decimal sumSquaredDiffXAndMean, decimal xSampleMean,bool isItPrediction)
        {
            var diff = x - xSampleMean;
            var p = isItPrediction ? 1 : 0;
            return t * s * (decimal)Math.Sqrt((double)(p +  1M / n + diff * diff /
sumSquaredDiffXAndMean));
        }


        private decimal GetPredictionInterval(decimal x)
        {
            return GetInterval(x, S, TDistribution, RawSample.Count,
SumSquaredDiffXAndMean, MeanNormalizedX, true);
        }


        private decimal GetConfidenceInterval(decimal x)
        {
            return GetInterval(x, S, TDistribution, RawSample.Count,
SumSquaredDiffXAndMean, MeanNormalizedX, false);
        }


        private bool DeleteOutliers()
        {
            if (HasRegressionBeenCalculated)
            {
                throw new ArgumentException("Regression has been calculated.
Outliers cannot be deleted again");
            }


            bool isThereAny = false;
            for (int i = 0; i < RawSample.Count; i++)
            {
                var predictionInterval = RegressionValues[i].PI;
                if (RawSample[i].Y > predictionInterval.UpperBound.Y ^
RawSample[i].Y < predictionInterval.LowBound.Y)
                {
                    isThereAny = true;
```

```
                RawSample.RemoveAt(i);
                RegressionValues.RemoveAt(i);
                i--;
            }
        }

        return isThereAny;
    }

    private bool IsThereOutliers()
    {
        VerifyCalculationState(false);

        for (int i = 0; i < RawSample.Count; i++)
        {
            var predictionInterval = RegressionValues[i].PI;
            if (RawSample[i].Y > predictionInterval.UpperBound.Y ^
RawSample[i].Y < predictionInterval.LowBound.Y)
            {
                return true;
            }
        }

        return false;
    }

    public  void  CalculateRegressionAtOnce()
    {
        VerifyCalculationState(false);

        do
        {
            OneStepOfTheCalculation();
        }
        while(DeleteOutliers());

        HasRegressionBeenCalculated = true;
    }
```

```
        private void OneStepOfTheCalculation()
        {
            var normalizedSample = GetNormalizedSample();
            CalculateAndSetB(normalizedSample);
            SetRegression(normalizedSample);
            SetIntervals(normalizedSample);
        }
        public bool CalculateRegressionByStep()
        {
            if (HasRegressionBeenCalculated) return false;

            if (!HasRegressionBeenCalculated && RegressionValues.Count != 0)
            {
                DeleteOutliers();
            }

            OneStepOfTheCalculation();
            HasRegressionBeenCalculated = !IsThereOutliers();

            return !HasRegressionBeenCalculated;
        }

        private void SetRegression(RelatedSample normalizedSample)
        {
            for (int i = 0; i < normalizedSample.Count; i++)
            {
                normalizedSample.Y[i] = Linear(normalizedSample.X[i]);
            }

            RegressionSample = (RelatedSample)normalizedSample.Clone();
            ReverseConversion(RegressionSample);
        }



        private decimal GetTDistribution(int degreesOfFreedomNumber)
        {
           return (decimal) new
Chart().DataManipulator.Statistics.InverseTDistribution(0.05,
degreesOfFreedomNumber);
        }
```

```
        private void SetIntervals(RelatedSample normalizedRegression)
        {
            RegressionValues.ClearAll();
            S = GetS(normalizedRegression, GetNormalizedSample());
            MeanNormalizedX = normalizedRegression.X.GetMean();
            SumSquaredDiffXAndMean =
normalizedRegression.X.GetSumSquaredDiffXAndSampleMean();
            TDistribution = GetTDistribution(normalizedRegression.Count - 2);

            for (int i = 0; i < normalizedRegression.Count; i++)
            {
                var confidence = GetConfidenceInterval(normalizedRegression.X[i]);

                confidence = (decimal)ReverseValue(confidence);


                var prediction = GetPredictionInterval(normalizedRegression.X[i]);

                prediction = (decimal)ReverseValue(prediction);

                RegressionValues.Add(new RegressionItem(RawSample.X[i],
                    (decimal) ReverseValue(normalizedRegression.Y[i]),
                    confidence,
                    prediction,
                    false));
            }
        }

        public RegressionItem GetRegressionYForArbitraryX(decimal x)
        {
            VerifyCalculationState(true);

            var normalizedX = (decimal)_logarithm((double)x);
            var normalizedY = Linear(normalizedX);
            var normalizedCI = GetConfidenceInterval(normalizedX);
            var normalizedPI = GetPredictionInterval(normalizedX);
            return new RegressionItem(x,
                (decimal) ReverseValue(normalizedY),
                (decimal) ReverseValue(normalizedCI),
```

```
            (decimal) ReverseValue(normalizedPI),
            true);
    }

    private void VerifyCalculationState(bool calculated)
    {
        if (HasRegressionBeenCalculated == !calculated)
            throw new ArgumentException("Regression calculation conflict");
    }

    public void AddRegressionYForArbitraryX(decimal x,bool needClearingBefore)
    {
        VerifyCalculationState(true);

        if (needClearingBefore) RegressionValues.ClearArbitrary();
        RegressionValues.Add(GetRegressionYForArbitraryX(x));
        RegressionValues.SortByX();
    }

    public decimal GetR2()
    {
        decimal sum = 0;

        for (int i = 0; i < RawSample.Count; i++)
        {
            var diff = RawSample.Y[i] - RegressionSample.Y[i];
            sum += diff * diff;
        }

        return 1 - sum / RegressionSample.Y.GetSumSquaredDiffXAndSampleMean();
    }

    public List<decimal> GetMRE()
    {
        var result = new List<decimal>();
        for (int i = 0; i < RawSample.Count; i++)
        {
            result.Add(Math.Abs((RawSample.Y[i] - RegressionSample.Y[i]) /
RawSample.Y[i]));
        }
```

```
            return result;
        }

        public decimal GetMMRE()
        {
            var mreList = GetMRE();
            return mreList.Sum() / mreList.Count;
        }

        public decimal GetPRED(decimal predicationOfLevel)
        {
            var mreList = GetMRE();
            var mreCountByCriterion = mreList.Count(t => t <= predicationOfLevel);
            return mreCountByCriterion / (decimal) mreList.Count;
        }
    }
}
```

Вміст файлу вихідного коду «Histogram.cs»:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

namespace CalcLOC
{
    class Histogram: IEnumerable<HistogramItem>
    {
        private readonly List<HistogramItem> _columns;
        public  int ColumnCount => _columns.Count;

        public decimal Delta { get; private set; }

        public decimal RandomVarsCount
        {
            get
            {
```

```
        return _columns.Sum(column => column.Height);
    }
}

internal HistogramEnum HistogramEnum
{
    get => default;
    set
    {
    }
}

internal HistogramItem HistogramItem
{
    get => default;
    set
    {
    }
}

public Histogram(List<decimal> rawData)
{
    _columns = new List<HistogramItem>();
    GenerateColumns(rawData.Count,rawData.Min(),rawData.Max());
    FillColumns(rawData);
}

private void FillColumns(List<decimal> rawData)
{
    foreach (var randomVar in rawData)
    {
        foreach (var column in _columns)
        {
            if (randomVar >= column.Start && randomVar <= column.End)
            {
                column.RandomVars.Add(randomVar);
                break;
            }
        }
    }
```

```
}

private void GenerateColumns(decimal count, decimal min, decimal max)
{
    var countColumns = CalculateCountColumns(count);
    var start = min;
    var delta = (max - min) / countColumns;

    for (decimal i = 0; i < countColumns; i++)
    {
        var hi = new HistogramItem();
        hi.Start = start + delta * i;
        hi.End = (i == countColumns - 1) ? max : start + delta * (i + 1);
        hi.Middle = (hi.Start + hi.End) / 2;
        _columns.Add(hi);
    }

    Delta = delta;
}

private static decimal CalculateCountColumns(decimal countOfRawData)
{
    var count= (decimal)(3.3 * Math.Log10((double)countOfRawData) + 1);
    return RoundCountColumns(count);
}

private static decimal RoundCountColumns(decimal count)
{
    decimal round = Decimal.Round(count);
    for (decimal i = 0; round % 2 == decimal.Zero; round += ++i)
    {
    }

    return round;
}

public IEnumerator<HistogramItem> GetEnumerator()
{
    return new HistogramEnum(_columns);
}
```

```
        IEnumerator IEnumerable.GetEnumerator()
        {
            return GetEnumerator();
        }
    }
}
```

Вміст файлу вихідного коду «Sample.cs»:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

namespace CalcLOC
{
    internal class Sample:
        IEnumerable<decimal>,
        ICloneable

    {
        private readonly List<decimal> _randomValues;

        public string Name { get; set; }

        public decimal Min => _randomValues.Min();
        public decimal Max => _randomValues.Max();

        public int Count => _randomValues.Count;

        public void Set(List<decimal> randomValues,bool dependent,string name)
        {
           Set(randomValues);
            Dependent = dependent;
            Name = name;
        }

        public void Set(List<decimal> randomValues)
        {
```

```
    _randomValues.Clear();
    _randomValues.AddRange(randomValues);
}

public void Add(decimal randomValue)
{
    _randomValues.Add(randomValue);
}

public void AddRange(List<decimal> randomValues)
{
    _randomValues.AddRange(randomValues);
}

public void RemoveAt(int index)
{
    _randomValues.RemoveAt(index);
}

public bool Dependent;

public Sample()
{
    Dependent = false;
    _randomValues = new List<decimal>();
}
public Sample(List<decimal> randomValues,bool dependent,string name):
    this()
{
    Set(randomValues,dependent,name);
}

public Histogram GetHistogram()
{
    return new Histogram(_randomValues);
}

public decimal GetMean()
{
    decimal mean = 0;
```

```
    var histogram = GetHistogram();
    foreach (var item in histogram)
    {
        mean += item.Middle * item.Height;
    }

    return mean / Count;
}

public decimal GetVariance()
{
    return GetCentralMoment(2);
}


public decimal GetStandardDeviation()
{
   return  (decimal)Math.Sqrt((double)GetVariance());
}

public bool IsNormal()
{
    return new PearsonTest(this).IsSampleNormal;
}

public PearsonTest GetPearsonTest()
{
    return new PearsonTest(this);
}

public decimal GetCentralMoment(uint n)
{
    var histogram = GetHistogram();
    var sampleMean = GetMean();
    decimal centralMoment = 0;
    foreach (var hi in histogram)
    {
        var diff = hi.Middle - sampleMean;
        decimal product = 1;
        for (int i = 0; i < n; i++)
```

```
                {
                    product *= diff;
                }
                centralMoment += product * hi.Height;
            }

            return centralMoment / Count;
        }

        public decimal GetSumSquaredDiffXAndSampleMean()
        {
            decimal sum = 0;
            decimal mean = GetMean();
            for (int i = 0; i < Count; i++)
            {
                var diff = _randomValues[i] - mean;
                sum += diff * diff;
            }

            return sum;
        }

        public decimal GetSkewness()
        {
            var m3 = GetCentralMoment(3);
            var standardDeviation = GetStandardDeviation();
            return m3 / (standardDeviation * standardDeviation *
standardDeviation);
        }

        public decimal GetKurtosis()
        {
            var standardDeviation = GetStandardDeviation();
            var sdDev4 = 1M;
            for (int i = 0; i < 4; i++) sdDev4 *= standardDeviation;
            return GetCentralMoment(4) / sdDev4 - 3;
        }

        public decimal this[int index]
        {
```

```
    get => _randomValues[index];
    set => _randomValues[index] = value;
}
public IEnumerator<decimal> GetEnumerator()
{
    return new SampleEnum(_randomValues);
}

IEnumerator IEnumerable.GetEnumerator()
{
    return GetEnumerator();
}

public decimal[] GetArray()
{
    return _randomValues.ToArray();
}



public List<decimal> GetCopyRandomValues()
{
    return new List<decimal>(GetArray());
}

public Sample GetNormalized(Func<double,double> log)
{
    var clone = (Sample)Clone();
    clone.Normalize(log);
    return clone;
}

public void Normalize(Func<double, double> log)
{
    for (int i = 0; i < Count; i++)
    {
        _randomValues[i] = (decimal)log((double)_randomValues[i]);
    }
}
public object Clone()
{
```

```
            var data = new List<decimal>();
            for (int i = 0; i < _randomValues.Count; i++)
            {
                data.Add(_randomValues[i]);
            }
            return new Sample(data, Dependent,Name);
        }

        public void Clear()
        {
            _randomValues.Clear();
        }

        public List<(string name, string val)> GetSampleProperties(int precision)
        {
            return new List<(string name, string val)>()
            {
                ("Count of data",Count.ToString()),
                ("Variance",$"{Math.Round(GetVariance(),precision):G29}"),
                ("Standard Deviation”,
$"{Math.Round(GetStandardDeviation(),precision):G29}"),
                ("Skewness",$"{Math.Round(GetSkewness(),precision):G29}"),
                ("Kurtosis",$"{Math.Round(GetKurtosis(),precision):G29}"),
                ("Mean",$"{Math.Round(GetMean(),precision):G29}"),
                ("Minimum",$"{Math.Round(Min,precision):G29}"),
                ("Maximum",$"{Math.Round(Max,precision):G29}"),
                ("Normal",IsNormal()?"Yes":"No")
            };
        }
    }
}
```

Вміст файлу вихідного коду «RInterval.cs»:

```
namespace CalcLOC
{
    class RInterval
    {
        public RPoint LowBound;
        public RPoint UpperBound;
```

```
        public RInterval(RPoint lowBound, RPoint upperBound)
        {
            LowBound = lowBound;
            UpperBound = upperBound;
        }
    }
}
```

Вміст файлу вихідного коду «RPoint.cs»:

```
namespace CalcLOC
{
    class RPoint
    {
        public decimal X;
        public decimal Y;

        public RPoint(decimal x, decimal y)
        {
            X = x;
            Y = y;
        }
    }
}
```

Вміст файлу вихідного коду «Calculation.cs»:

```
using System;
using System.Windows.Forms;

namespace CalcLOC
{
    public class CalculationForm : Form
    {
        private NumericUpDown numericUpDownNC;
        private Button buttonCalculate;
        private Label labelNC;
```

```
private Label labelLOC;
private Label labelCI;
private Label labelPI;

public CalculationForm ()
{
    this.Text = "Calculation";
    this.Width = 300;
    this.Height = 250;

    // NumericUpDown для NC
    numericUpDownNC = new NumericUpDown
    {
        Location = new System.Drawing.Point(10, 10),
        Minimum = 1,
        Maximum = 1000,
        Value = ""
    };
    this.Controls.Add(numericUpDownNC);

    var ncLabel = new Label
    {
        Text = "NC",
        Location = new System.Drawing.Point(120, 12),
        AutoSize = true
    };
    this.Controls.Add(ncLabel);

    buttonCalculate = new Button
    {
        Text = "Calculate",
        Location = new System.Drawing.Point(10, 45),
        Width = 260
    };
    buttonCalculate.Click += ButtonCalculate_Click;
    this.Controls.Add(buttonCalculate);

    labelNC = CreateLabel("NC: ", 80);
    labelLOC = CreateLabel("LOC: ", 100);
    labelCI = CreateLabel("Confidence Interval: ", 120);
```

```
    labelPI = CreateLabel("Prediction Interval: ", 140);
}

private Label CreateLabel(string text, int top)
{
    var label = new Label
    {
        Text = text,
        Location = new System.Drawing.Point(10, top),
        AutoSize = true
    };
    this.Controls.Add(label);
    return label;
}

private void ButtonCalculate_Click(object sender, EventArgs e)
{
    int nc = (int)numericUpDownNC.Value;

    int loc = (decimal)ReverseValue(nc);
    GetPredictionInterval((decimal)loc);
    GetConfidenceInterval((decimal)loc);

    labelNC.Text = $"NC: {nc}";
    labelLOC.Text = $"LOC: {loc}";
    labelCI.Text = $"Confidence Interval: [{ciLow}, {ciHigh}]";
    labelPI.Text = $"Prediction Interval: [{piLow}, {piHigh}]";
}
```

Додаток В

Опис програми для оцінювання метрики LOC програмних застосунків на C++

1 Загальні відомості

1.1 Найменування і позначення програми

Найменування: Програма для оцінювання метрики LOC програмних застосунків на C++.

Позначення: «CalcLOC».

1.2 Програмне забезпечення, необхідне для функціонування програми

Для повноцінного функціонування програми необхідно використовувати ОС Windows версії не нижче 7, .Net Framework 4.8.1.

1.3 Мови програмування

Програмне забезпечення «CalcLOC» розроблено мовою програмування C#.

2 Функціональне призначення

Програмне забезпечення «CalcLOC» створено для спрощення розрахунку оцінки метрики LOC програмних застосунків на C++ і візуалізація результатів розрахунку.

Програма виконує наступні функції:

розрахувати числові характеристики вибірки, такі, як: кількість значень, вибіркове середнє, дисперсія, асиметрія, ексцес, результат тесту Пірсона на нормальність розподілу випадкової величини;

побудувати гістограми вихідної вибірки та графік з її точками;

– розрахувати числові характеристики регресії, такі, як: коефіцієнти рівняння регресії (b_0 та b_1), коефіцієнт детермінації (R_2), середнє значення відносної похибки (*MMRE*), відсоток значень, які мають відносну похибку 0,25 (*PRED*(0,25));

розрахувати значення границь довірчого інтервалу та інтервалу прогнозування;

побудувати графік регресії, довірчого інтервалу та інтервалу прогнозування;

розрахувати значення метрики LOC за введеним значенням кількості класів;

переглянути точки отриманої вибірки, представлені числовими значеннями;

- переглянути розраховані точки регресії, довірчого інтервалу та інтервалу прогнозування у вигляді числових значень.

3 Виклик та завантаження

Запуск програмного забезпечення «CalcLOC» приводиться запуском файлу з назвою CalcLOC.exe.

4 Вхідні дані

Для розрахунку регресії введення даних здійснюється із файлу формату .csv, який містить наступні стовпці: значення кількості класів NC, значення метрики LOC.

Для прогнозування значення метрики LOC згідно регресії введення даних здійснюється до відповідного поля вводу вручну з клавіатури.

Вхідні дані:

для розрахунку регресії: кількість класів, LOC;

для прогнозування значення метрики LOC згідно регресії: кількість класів.

5 Вихідні дані

числові характеристики вибірки, такі, як: кількість значень, вибіркове середнє, дисперсія, асиметрія, ексцес, результат тесту Пірсона на нормальність розподілу випадкової величини;

гістограми вибірки;

відповідні точки вибірки;

числові характеристики регресії, такі, як: коефіцієнти рівняння регресії (b_0 та b_1), коефіцієнт детермінації (R_2), середнє значення відносної похибки (*MMRE*), відсоток значень, які мають відносну похибку 0,25 (*PRED*(0,25));

графік, який відображає лінію регресійної функції, довірчий інтервал, інтервал прогнозування, точки даних;

значення метрики LOC;

відображення на графіку моделі розрахованої точки.

Додаток Г

Інструкція користувача програми для оцінювання метрики LOC програмних застосунків на C++

Для запуску програми треба зайти в папку програми та запустити файл з назвою CalcLOC.exe.

При запуску файлу з назвою CalcLOC.exe відкриється головне вікно програми.

Розрахувати числові характеристики вихідної вибірки

Для розрахунку числових характеристик вихідної вибірки треба:

– перейти на вкладку «Histograms»;
– в пункті меню «Sample» обрати підпункт «Original».

На вкладці «Histograms» будуть зображені гістограми вибірки та їх числові характеристики.

Розрахувати числові характеристики нормалізованої вибірки десятковим логарифмом

Для розрахунку числових характеристик нормалізованої вибірки десятковим логарифмом треба:

– перейти на вкладку «Histograms»;
– в пункті меню «Sample» обрати підпункт «Logarithm 10».

На вкладці «Histograms» будуть зображені гістограми вибірки та їх числові характеристики.

Розрахувати значення метрики LOC

Для розрахунку значення метрики LOC треба:

– перейти на вкладку «Regression»;
– у блоці «Calculation», в полі «NC» ввести значення кількості класів;

– натиснути кнопку «Calculate».

У блоці «Calculation» буде виведено розрахунки значення метрики LOC та границь довірчого інтервалу та інтервалу прогнозування (див. рис. Г.1)

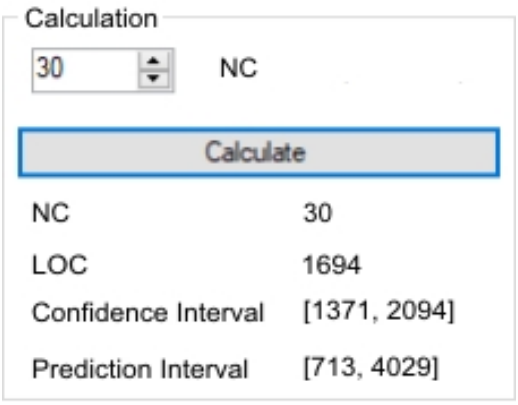


Рисунок Г.1 – Блок «Calculation» з розрахунком значення метрики LOC та границь довірчого інтервалу та інтервалу прогнозування

Додаток Д

Програма та методика випробування програми для оцінювання метрики LOC програмних застосунків на C++

1 Об'єкт випробування

Об’єктом випробування є програма для оцінювання метрики LOC програмних застосунків на C++.

2 Мета випробування

Метою проведення випробування є перевірка працездатності даного програмного продукту, перевірка відповідності характеристик та вимог розробленої програми, викладених у технічному завдані.

3 Вимоги до додатка

При проведенні випробувань функціональні характеристики програми підлягають перевірці на відповідність вимогам, викладеним у пункті «Вимоги до функціональних характеристик» технічного завдання.

4 Вимоги до программы

4.1 Склад програмної документації, пропонованої на випробування

Програмна документація повинна містити наступні документи:

- технічне завдання;
- опис програми;
- код програми;
- інструкція користувача;
- програма та методика випробувань програми.

4.2 Спеціальні вимоги

Спеціальні вимоги до програмної документації не висуваються.

5 Засоби та порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Склад технічних засобів, що використовувалися під час випробувань повинен бути не нижче, ніж вказано у пункті «Вимоги до складу та параметрів технічних засобів» технічного завдання.

5.2 Програмні засоби, що використовуються під час випробувань

Система повинна задовольняти вказаним мінімальним вимогам, вказаним у пункті «Вимоги до інформаційної та програмної сумісності» технічного завдання.

5.3 Порядок проведення випробувань

Порядок проведення випробувань складається з перевірки вимог до функціональних характеристик програмного продукту та перевірки вимог до програмної документації.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до програмної документації

Перевірка дотримання вимог програмної документації на програмний продукт виконується візуально представником служби, відповідальної за експлуатацію. У ході перевірки зіставляється склад і комплектність програмної документації, представленої розробником, з переліком програмної документації, наведеним у пункті «Склад програмної документації, пропонованої на випробування» цього документа.

Перевірка вважається завершеною у випадку відповідності складу та комплектності програмної документації, представленої розробником, переліку

програмної документації, наведеному у зазначеному вище пункті.

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Перевірка працездатності програми виконується згідно з пунктом «Вимоги до функціональних характеристик» технічного завдання.

Перевірка вважається завершеною у разі відповідності складу і послідовності дій при виконанні даної перевірки, вказаною вище пункту технічного завдання.

В процесі тестування буде перевірена функціональність програми для оцінювання метрики LOC програмних застосунків на C++.

6.3 Методика випробувань основних функціональних можливостей

6.3.1 Перевірка дії «Перший запуск програми»

Для перевірки запуску програми потрібно запустити файл з назвою CalcLOC.exe.

Очікуваний результат: головне вікно програми, показане на екрані монітора.

6.3.2 Перевірка дії «Нормалізувати дані десятковим логарифмом»

Для перевірки нормалізування даних десятковим логарифмом треба відкрити вкладку «Histograms». У пункті головного меню «Sample» обрати пункт «Logarithm 10».

Очікуваний результат: на вкладці «Histograms» відобразяться наступні дані:

- числові характеристики нормалізованих даних кількості класів;
- гістограма нормалізованих даних кількості класів;
- числові характеристики нормалізованих даних метрики LOC;
- гістограма нормалізованих даних метрики LOC;
- точки нормалізованих даних вибірки.

6.3.3 Перевірка дії «Розрахувати значення метрики LOC»

Для перевірки розрахунку значення метрики LOC треба на вкладці «Regression», у блоці «Calculation» в полі «NC» ввести значення кількості класів програмного забезпечення та натиснути кнопку «Calculate».

Очікуваний результат: у блоці «Calculation» будуть показані значення метрики LOC та границь довірчого інтервалу та інтервалу прогнозування.