МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка програмного забезпечення для знаходження внесених змін в код виконуваних файлів

Виконав: студент групи 4151

_____ Камінський С.С.

(підпис, ПІБ)

Керівник роботи:

доцент кафедри ПЗАС, к.т.н., доцент

(посада, науковий ступень вчене звання)

_____ Макарова Л.М.

(підпис, ПІБ)

Миколаїв – 2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

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

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«___» _____ 2023 р.

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

Студенту: Камінському Станіславу Станіславовичу _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного забезпечення для знаходження внесених змін в код виконуваних файлів

Керівник роботи доцент кафедри ПЗАС, к.т.н., доцент Макарова Л.М. _____

Затверджені наказом ректора №257-уч від «17» березня 2023 р. _____

2. Термін подання роботи: 29.05.2023 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, Аналіз вимог до ПЗ, Архітектура ПЗ, Діаграма варіантів використання, Технічний проект ПЗ, Вибір інструментів розробки, Реалізація та Тестування ПЗ, Результати розробки, 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 20.03.2023 р. _____

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	24.03.2023	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	31.03.2023	ПП*
3.	Аналіз вимог до ПЗ	07.04.2023	ПП*
4.	Розробка архітектури ПЗ	21.04.2023	
5.	Розробка проекту ПЗ	05.05.2023	
6.	Реалізація та тестування ПЗ	12.05.2023	
7.	Підготовка розділу Результати розробки ПЗ	17.05.2023	
8.	Підготовка розділу з охорони праці	19.05.2023	ПП*
9.	Підготовка розділу ВИСНОВКИ	24.05.2023	
10.	Оформлення списку використаних джерел та додатків	26.05.2023	
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.)	29.05.2023	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
12.	Підготовка презентації та доповіді	02.06.2023	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	05.06.2023	
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	12.06.2023	

* - за результатами переддипломної практики (ПП), яка була з 01.02.2023 до 19.03.2023 р.

Студент _____ Камінський С.С.
(підпис) (ПІБ)

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

АНОТАЦІЯ

Кваліфікаційна робота присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці програмного забезпечення для знаходження внесених змін в код виконуваних файлів

Об'єктом даної роботи є процес розробки програмного забезпечення, призначеного для знаходження внесених змін в код виконуваних файлів. В рамках кваліфікаційної роботи був проведений аналіз практичної задачі інженерії програмного забезпечення, включаючи вивчення існуючих аналогів та методів, що використовуються для виявлення змін у коді виконуваних файлів. Здійснено постановку задачі на розробку програмного забезпечення для знаходження внесених змін в код виконуваних файлів. Було розроблено проєкт програмного забезпечення для знаходження внесених змін в код виконуваних файлів. Проведено опис аспектів охорони праці при роботі з екранними пристроями.

Кваліфікаційна робота викладена на 99 сторінках друкованого тексту, містить 28 рисунків, 9 таблиць, 5 додатків та список використаних джерел із 24 найменувань.

ABSTRACT

The qualification work is devoted to solving a practical problem of software engineering, characterized by complexity and uncertainty of conditions, namely the development of software for detecting changes made to the code of executable files

The object of this work is the process of developing software designed to detect changes made to the code of executable files. As part of the qualification work, an analysis of the practical problem of software engineering was carried out, including the study of existing analogues and methods used to detect changes in the code of executable files. The task of developing software for detecting changes made to the code of executable files was set. A software project was developed to detect changes made to the code of executable files. The aspects of labor protection when working with screen devices were described.

The qualification work is presented on 99 pages of printed text, contains 28 figures, 9 tables, 5 appendices and a list of references of 24 titles.

ЗМІСТ

ВСТУП..... 7
1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗНАХОДЖЕННЯ ВНЕСЕНИХ ЗМІН В КОД ВИКОНУВАНИХ ФАЙЛІВ 9
1.1 Аналіз практичної задачі інженерії програмного забезпечення 9
1.2 Аналіз існуючих аналогів та обґрунтування необхідності розробки програмного забезпечення для знаходження внесених змін в код виконуваних файлів..... 11
1.3 Постановка задачі на розробку програмного забезпечення для знаходження внесених змін в код виконуваних файлів 13
2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗНАХОДЖЕННЯ ВНЕСЕНИХ ЗМІН В КОД ВИКОНУВАНИХ ФАЙЛІВ 16
2.1 Аналіз вимог до програмного забезпечення для знаходження внесених змін в код виконуваних файлів..... 16
2.1.1 Побудова моделі варіантів використання програмного забезпечення для знаходження внесених змін в код виконуваних файлів..... 16
2.1.2 Специфікації варіантів використання програмного забезпечення для знаходження внесених змін в код виконуваних файлів 18
2.2 Архітектура програмного забезпечення для знаходження внесених змін в код виконуваних файлів 22
2.2.1 Розробка діаграми компонентів програмного забезпечення для знаходження внесених змін в код виконуваних файлів..... 23
2.2.2 Розробка діаграми розміщення програмного забезпечення для знаходження внесених змін в код виконуваних файлів..... 27
2.3 Проект програмного забезпечення для знаходження внесених змін в код виконуваних файлів 28

2.3.1 Ескізний проект програмного забезпечення для знаходження внесених змін в код виконуваних файлів 28
2.3.2 Технічний проект програмного забезпечення для знаходження внесених змін в код виконуваних файлів 38
2.3.3 Робочий проект програмного забезпечення для знаходження внесених змін в код виконуваних файлів 49
3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗНАХОДЖЕННЯ ВНЕСЕНИХ ЗМІН В КОД ВИКОНУВАНИХ ФАЙЛІВ 53
4 ОХОРОНА ПРАЦІ 59
4.1 Аналіз шкідливих та небезпечних факторів під час роботи з екранними пристроями 60
4.2 Методи нейтралізації або зменшення впливу негативних факторів під час роботи з екранними пристроями 63
ВИСНОВКИ 65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 66
ДОДАТОК А – Технічне завдання на розробку програмного забезпечення для знаходження внесених змін в код виконуваних файлів 69
ДОДАТОК Б – Вихідні тексти програмних модулів 76
ДОДАТОК В – Опис програмного забезпечення 88
ДОДАТОК Г – Інструкція користувача програмного забезпечення для знаходження внесених змін в код виконуваних файлів 90
ДОДАТОК Д – Програма та методика випробувань програмного забезпечення .. 97

ВСТУП

Науково-практичні досягнення в області інформаційних технологій дозволяють розробляти нові програмні продукти з різними функціональними можливостями та забезпечувати їх високу якість. Однак, разом зі зростанням складності програмних продуктів, збільшується кількість чинників, які впливають на їх роботу та вимагають уваги розробників [1].

Один з таких чинників – зміна вихідного коду програмного забезпечення [2], що може статися через низку причин, таких як внесення змін розробниками, вірусні атаки та інші. У такому випадку важливо здійснювати моніторинг програмного коду для виявлення будь-яких змін та оцінки їх впливу на функціональність програми.

Тому метою цієї роботи є розробка програмного забезпечення DisEn (Disassembly and Entropy Analysis), яка дозволяє аналізувати програми на основі дисасемблювання та визначення ентропії їхнього вихідного коду. Ця програма дозволить виявляти зміни в коді програми та порівнювати їх з попередніми версіями.

Розробка програмного забезпечення з різними функціональними можливостями є складною задачею, яка вимагає уваги до деталей та врахування багатьох чинників. Аналіз програмного коду з використанням інструментів, таких як DisEn, дозволяє забезпечити високу якість програмного забезпечення та запобігти можливим проблемам у майбутньому.

Розробка ПЗ для аналізу виконуваних файлів є дуже важливою задачею [3]. На сьогоднішній день існує декілька аналогів, таких як IDA Pro, OllyDbg, Hopper та інші. Проте, багато з цих програм мають платну ліцензію або обмежений функціонал. Також важливою перевагою розробленої програми є можливість порівняння двох версій файлів із збереженням попередньої версії в якості базової.

Отже, розробка нової програми, яка вирішує задачу аналізу виконуваних файлів з використанням підрахунку ентропії та порівнянням версій файлів, є дуже доцільною і має багато переваг порівняно з існуючими аналогами.

Основним завданням кваліфікаційної роботи є розробка програмного забезпечення для аналізу виконуваних файлів на наявність змін та визначення можливої авторської інтервенції чи внесення шкідливого коду.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– Розробити процедуру для обробки даних у виконуваних файлів.

– Розробити процедуру для підрахунку ентропії команд виконуваних файлів.

– Розробити процедуру для порівняння ентропії команд виконуваних файлів.

– Розробити інтерфейс користувача для відображення результатів аналізу виконуваних файлів.

– Розробити процедуру для зберігання даних про попередню версію файлу.

– Розробити процедуру для порівняння двох версій файлу і відображення різниці в ентропії команд, кількості команд та розмірі файлу.

– Розробити процедуру для побудови графік ентропії команд виконуваних файлів.

Всі вищезазначені завдання повинні бути вирішені з урахуванням сучасних технологій програмування та аналізу даних.

Об'єктом кваліфікаційної роботи є процес розробки програмного забезпечення для знаходження внесених змін в код виконуваних файлів.

Всі ці завдання допоможуть досягти поставленої мети переддипломної практики та вдосконалити вміння та навички у галузі розробки програмного забезпечення.

Результати кваліфікаційної роботи висвітлено у 3 наукових працях – тезах конференцій [4, 5] та фаховій статті [6].

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗНАХОДЖЕННЯ ВНЕСЕНИХ ЗМІН В КОД ВИКОНУВАНИХ ФАЙЛІВ

1.1 Аналіз практичної задачі інженерії програмного забезпечення

Аналіз практичної задачі інженерії програмного забезпечення, пов'язаної з розробкою програмного забезпечення "DisEn", стикається зі складними вимогами та викликами. "DisEn" є інструментом для аналізу та виявлення чужорідного коду виконуваних файлів.

Основна складність полягає у принципі аналізу виконуваних файлів щодо знаходження чужорідного коду. Для цього будемо використовувати значення метрики ентропії.

Для аналізу коду використовується розрахунок ентропії, що дозволяє проаналізувати файли та відрізняти їх по результатам обчислення.

Інформаційна ентропія – міра невизначеності або непередбачуваності інформації, невизначеність появи будь-якого символу первинного алфавіту. За відсутності інформаційних втрат ентропія чисельна дорівнює кількості інформації на символ переданого повідомлення. Наприклад, у послідовності букв, що складають будь-яке речення українською мовою, різні букви з'являються з різною частотою, тому невизначеність появи для деяких букв менша, ніж для інших. Якщо врахувати, що деякі поєднання букв зустрічаються дуже рідко, то невизначеність зменшується ще сильніше.

Ентропія також може бути застосована до програмного забезпечення, як міра взаємодії. Так, розглядаючи ентропію вихідного коду, який може бути розбитий на більш дрібні сегменти (рядки, функції, змінні, команди і т.п.), можна визначати авторство тієї чи іншої [7] частини коду. Таким чином, можна обчислити внесок

різних авторів у розроблену програму. Для розрахунку ентропії в теорії інформації використовують таку формулу:

$$H(i) = -p_i * log_2(p_i)\ , \qquad (1)$$

де i – можливі стани,

p_i – імовірність появи i–го стану.

У випадку з програмним кодом у ролі станів виступають команди, і, відповідно, підраховується ймовірність їх появи в коді:

$$p_i = n_i / N,$$

де n_i – кількість появи i-ї команди,

N – загальна кількість команд у файлі.

Деякі метрики програмного забезпечення, наприклад, метрики Холстеда чи Берлінгера, беруть за основу ентропію. Також ентропійний метод використовується для вирішення таких завдань, як пошук зашифрованого або запакованого шкідливого програмного забезпечення.

Одним із способів виявлення авторства змін у тексті, зокрема, і комп`ютерної програми, є застосування кількісної метрики – ентропії, що добре підходить для визначення авторства програмного коду, тобто, використовуючи дану метрику, можна з високою ймовірністю стверджувати, написана нова ділянка коду автором програми чи є чужою вставкою.

Інформаційна ентропія Шеннона [8] була обрана як основа для розрахунку завдяки тому, що вона дає змогу отримувати кількісні результати, які зручно порівнювати один з одним і, тим самим, визначати авторство зміни файлу.

Елементами для розрахунку метрики виступають асемблерні команди виконуваних файлів. Тож, щоб отримати ці команди, треба спочатку дизасемблювати файл.

1.2 Аналіз існуючих аналогів та обґрунтування необхідності розробки

В першу чергу були розглянуті такі існуючі рішення для дизасемблювання, як IDA Pro Disassembler [9] та Immunity Debugger [10].

IDA Pro Disassembler [9] – інтерактивний дизасемблер (див. рис 1.1), який широко використовується для реверс-інжинірингу. Він відрізняється винятковою гнучкістю, наявністю вбудованої командної мови, підтримує безліч форматів файлів для великої кількості процесорів і операційних систем.

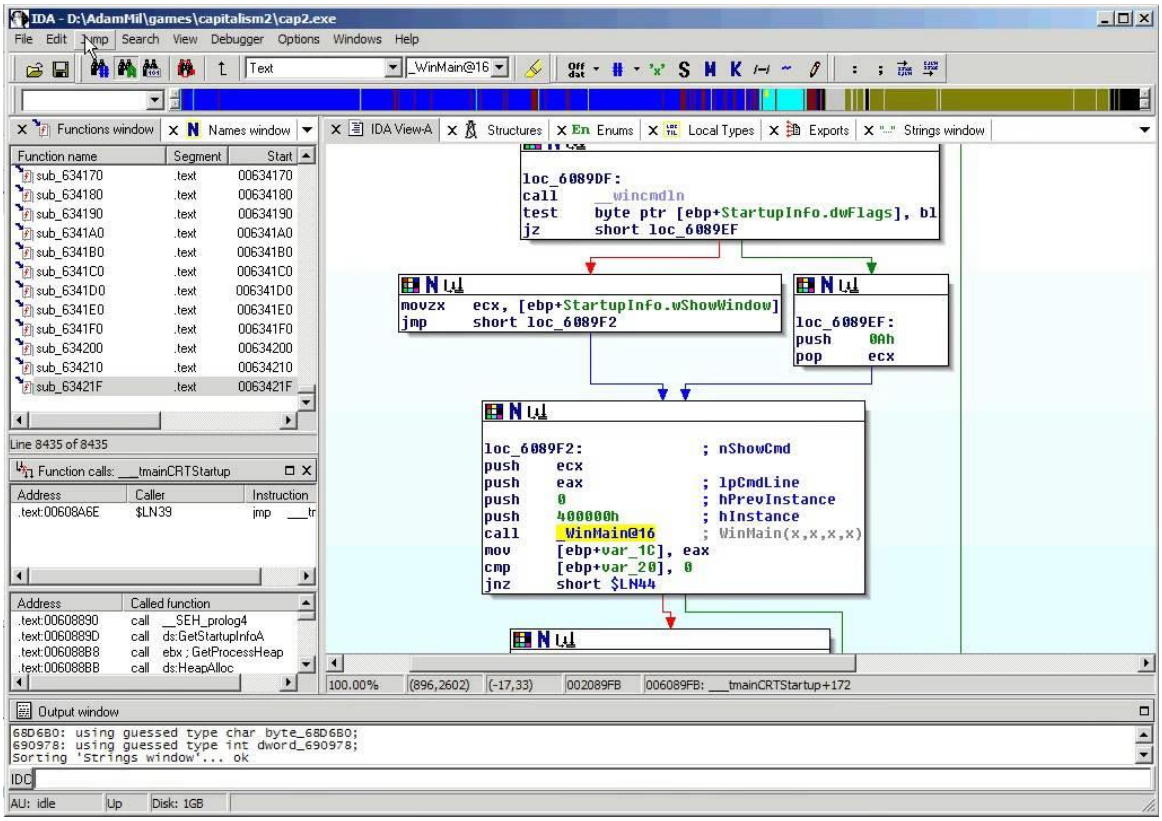

Рисунок 1.1 – Зовнішний вигляд IDA Pro Disassembler [9]

Immunity Debugger [10] – це новий потужний спосіб написання експлойтів (див. рис 1.2), аналізу шкідливих програм та аналізу бінарних файлів. Він заснований на надійному інтерфейсі користувача з графічними функціями. Розроблений спеціально для створення множин з великим і добре підтримуваним інтерфейсом Python API для легкої розширюваності.

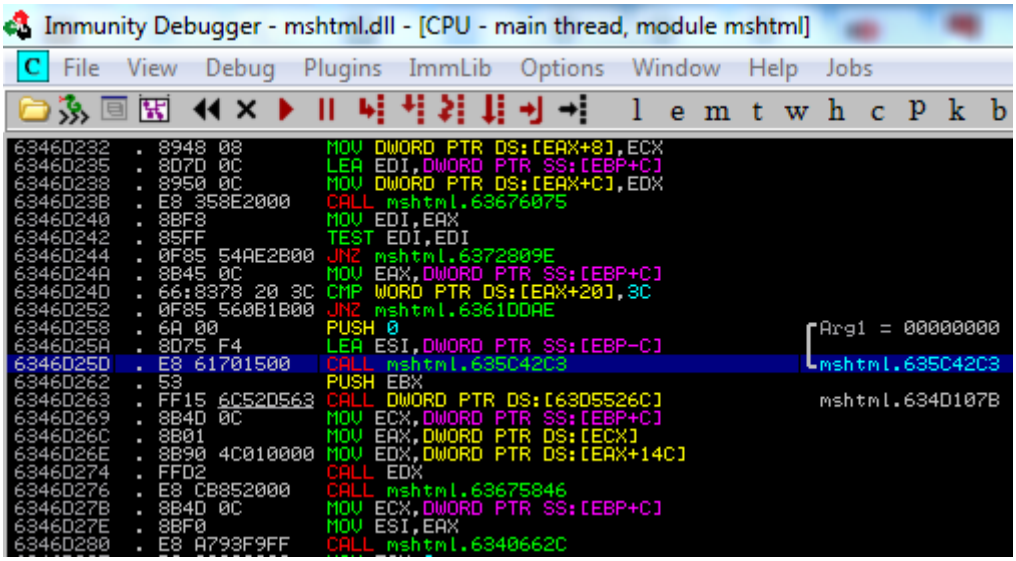

Рисунок 1.2 – Зовнішний вигляд Immunity Debugger [10]

Однак вищеназвані дизасемблери надто важкі і великі. Тому були знайденні більш легкі там швидкі аналоги.

Capstone [11] – це легкий мультиплатформенний фреймворк для дизасемблювання. Його головна особливість – це мультиархітектурність: Arm, Arm64 (Armv8), BPF, Ethereum Virtual Machine, M68K, M680X, Mips, MOS65XX, PowerPC, RISCV, Sparc, SystemZ, TMS320C64X, Web Assembly, XCore & X86 (включаючи X86_64).

Але даний дизасемблер також не відповідає потребам, оскільки в результаті його роботи не повертається повний набір команд. Крім того, він має обмеження на розмір виконуваних файлів та може займати багато часу для обробки файлів розміром більше 5 Мб, а іноді призводить до аварійного завершення програми. Незважаючи на ці недоліки, даний дизасемблер може працювати з будь-яким типом файлів.

Інший дизасемблер, який був розглянутий, є утилітою, що йде разом з Visual Studio 2019 [12]. Програма дампу двійкових файлів DUMPBIN.EXE [13] відображає двійкові файли у форматі текстового файлу із командами асемблера. Вона не має обмежень на розмір вхідного виконуваного файлу і швидко їх дизасемблює, але вона може працювати лише с файлами типу .exe. Проте, цей дизасемблер повністю

виконує потрібні задачі, тому саме він став основним інструментом дизасемблювання виконуваних файлів.

Після проведення [14] аналізу практичної задачі інженерії програмного забезпечення та виявлення недоліків існуючих аналогів, було вирішено розробити нове програмне забезпечення, яке спроможне допомогти визначати авторство змін у коді виконуваних файлів. Це програмне забезпечення отримало назву "DisEn".

1.3 Постановка задачі на розробку програмного забезпечення

Метою даної роботи є розробка програмного забезпечення "DisEn" для надання зручної та ефективної платформи для виявлення змін у коді виконуючих файлах. Головною метою є задоволення потреб користувачів і сприяння забезпеченню безпеки програмного забезпечення.

Експлуатаційне призначення:

Експлуатаційним призначенням програмного забезпечення «DisEn» є пошук та виявлення внесених змін в код виконуваних файлах.

Функціональне призначення:

Функціональним призначенням програмного забезпечення «DisEn» є покращення та полегшення процесу пошуку та виявлення внесених змін в код виконуваних файлах.

Функції користувача:

– Аналіз коду – користувач має можливість завантажити виконуваний файл для подальшого аналізу авторства змін, а саме : дизасемблювання файлу с можливістю перегляду тексту отриманого вихідного коду виконуваного файлу і розрахунок ентропії згідно з формулою інмовфрності.

– Перегляд результатів – користувач може переглянути результати аналізу коду дизасамбльованного виконуваного файлу, а саме розраховані на попередньому кроку значення, переглянути отримані результати у табличній формі де містиця інформація про команди їх кількість та ентропію, та у формі графіку.

– Збереження результатів – користувач має можливість зберегти результати аналізу для подальшого використання або архівування.

Вимоги до організації вхідних та вихідних даних:

Вхідні дані:

– Вхідними даними для програми є виконуваний файл, який підлягає дизасемблюванню файлу с можливістю перегляду тексту отриманого вихідного коду виконуваного файлу і розрахунок ентропії згідно з формулою Шеннона. Виконуваний файл повинен бути файлом .exe формату, який підтримується програмою DisEn.

– Для забезпечення коректного аналізу, виконуваний файл повинен мати повністю завантажені дані та бути не пошкоджений.

Вихідні дані:

– Результатом роботи програми є аналітичний звіт, який містить інформацію про ентропію команд дізасемблірованого виконавчого файлу. Якщо файл був вже аналізований програмним забезпеченням, є можливість отримати порівняння поточної та попередньої версії цього файлу. Звіт може бути збережений у текстовому або графічному форматі.

– Вихідні дані подаються у вигляді файлу формату .txt з вихідним кодом дизасембльованного виконуваного файлу, результатом є таблиця, котра має наступні поля: команда, кількість команд, ентропія. Також результати можуть бути представленні у вигляді графіка.

Вимоги до складу та параметрів технічних засобів:

Для роботи з програмним забезпеченням DisEn рекомендовані наступні мінімальні параметри технічних засобів:

– Процесор Intel Core i5 або еквівалентного рівня

– Оперативна пам'ять 4 ГБ.

– Дискового простору 100 МБ для встановлення програми та збереження результатів аналізу.

– Роздільна здатність екрану не менше 1280x800 пікселів.

3. Вимоги до інформаційної та програмної сумісності:

– Версія операційної системи Windows повинна підтримувати WPF та .NET Framework 4.5 або вище.

– Рекомендована версія операційної системи не нижче Windows 7.

Більш докладно постановка задачі сформульована у технічному завданні, яке наведене у Додатку А.

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «DisEn» ДЛЯ ЗНАХОДЖЕННЯ ВНЕСЕНИХ ЗМІН В КОД ВИКОНУВАНИХ ФАЙЛІВ

2.1 Аналіз вимог до програмного забезпечення для знаходження внесених змін в код виконуваних файлів

2.1.1 Побудова моделі варіантів використання програмного забезпечення для знаходження внесених змін в код виконуваних файлів

Під час аналізу предметної галузі та визначення вимог був виявлений основний користувач програмного забезпечення для виявлення внесених змін в код виконуваних файлів. Цей користувач, якого ми називаємо "Користувачем", взаємодіє з програмою та виконує роль в основних сценаріях використання. Він використовує програму для аналізу коду та виявлення внесених змін. Користувач відповідає за введення вхідних даних, вибір необхідних параметрів та отримання результатів аналізу. Детальніші відомості про цього користувача можна знайти у таблиці 2.1.

Таблиця 2.1 – Виявлені актори

Актор	Короткий опис
Користувач	Користувач програмного забезпечення є основним актором програмного забезпечення. Він взаємодіє з програмою для виявлення внесених змін в код виконуваних файлів. Користувач відповідає за надання вхідних даних, встановлення параметрів аналізу та отримання результатів. Він має розуміння процесу роботи з програмним забезпеченням та має необхідні навички для ефективного використання програми "DisEn".

Виявлені варіанти використання програмного забезпечення «DisEn» для знаходження внесених змін в код виконуваних файлів зведено до таблиці 2.2.

Таблиця 2.2 – Виявлені варіанти використання програмного забезпечення «DisEn» для знаходження внесених змін в код виконуваних файлів

Актор	Назва	Короткий опис
Користувач	Дизасемблювання виконуваного файлу	Користувач запускає "DisEn" для аналізу виконуваних файлів і виявлення внесених змін у коді.
Користувач	Перегляд вихідного тексту	Користувач використовує "DisEn" для перегляду вихідного тексту виконуваного файлу.
Користувач	Розрахунок ентропії	Користувач використовує "DisEn" для розрахунку ентропії виконуваного файлу.
Користувач	Представлення результатів у вигляді таблиці	Користувач використовує "DisEn" для представлення результатів розрахунку у вигляді таблиці.
Користувач	Представлення результатів у вигляді графіка	Користувач використовує "DisEn" для представлення результатів розрахунку у вигляді графіка.
Користувач	Порівняння з попередньою версією	У випадку наявності попередньої версії програми, користувач використовує "DisEn" для порівняння з нею.

На рисунку 2.1 представлено діаграму варіантів використання.

Рисунок 2.1 – Діаграма варіантів використання програмного забезпечення для знаходження внесених змін в код виконуваних файлів

Наведений вище рисунок побудови діаграми варіантів використання дозволить розробити специфікації.

2.1.2 Специфікації варіантів використання програмного забезпечення для знаходження внесених змін в код виконуваних файлів

Специфікації варіантів використання програмного забезпечення представлені в таблицях 2.3 – 2.8.

Таблиця 2.3 – Специфікація використання «Дизасемблювання виконуваного файлу»

Складова	Опис
Опис прецеденту	Дизасемблювання виконуваного файлу
Дійові особи	Користувач
Передумови	Виконуваний файл існує
Потік подій	1. Користувач відкриває програму для дизасемблювання виконуваного файлу. 2. Користувач вказує шлях до виконуваного файлу, який потрібно дизасемблювати. 3. Програма зчитує виконуваний файл. 4. Програма аналізує бінарний код виконуваного файлу та перетворює його на асемблерний код.
Базовий потік	1. Користувач вказує шлях до виконуваного файлу, який потрібно дизасемблювати. 2. Програма зчитує виконуваний файл. 3. Програма аналізує бінарний код виконуваного файлу та перетворює його на асемблерний код.
Альтернативні потоки	1. Користувач не відкриває програму для дизасемблювання виконуваного файлу. 2. Користувач не вказує шлях до виконуваного файлу. Процес не починається. 3. Виникає помилка при зчитуванні виконуваного файлу. Процес аналізу не відбувається.
`Постумови	Програма успішно дизасемблювала виконуваний файл і згенерувала асемблерний код. Користувач може переглянути асемблерний код та використати його для подальшого вивчення або аналізу програми, яка створила виконуваний файл.

Таблиця 2.4 – Сценарій використання «Перегляд вихідного тексту»

Складова	Опис
Опис прецеденту	Перегляд вихідного тексту
Дійові особи	Користувач
Передумови	Дизасемблювання виконуваного файлу завершено, отримано файл формата «. txt» з вихідним кодом
Потік подій	1. Користувач відкриває файл формата «. txt» у текстовому редакторі з отриманим вихідним кодом. 2. Користувач переглядає вихідний код.
Базовий потік	1. Користувач відкриває файл формата «. txt» у текстовому редакторі з отриманим вихідним кодом. 2. Користувач переглядає вихідний код.
Альтернативні потоки	1.Користувач не має отриманого txt файлу з вихідним кодом. Перегляд не може початись. 2.Під час перегляду вихідного коду сталася помилка або програма зупинилась. Перегляд не може продовжуватись.
Постумови	Користувач успішно переглядає вихідний код і здійснює потрібні маніпуляції з текстом за допомогою програми або текстового редактора. Користувач може зберегти зміни, використати код для подальшої роботи або закрити файл.

Таблиця 2.5 – Специфікація використання «Розрахунок ентропії»

Складова	Опис
1	2
Опис прецеденту	Розрахунок ентропії
Дійові особи	Користувач
Передумови	Виконуваний файл був дизасембльований і отриманий вихідний код виконуваного файлу
Потік подій	1. Користувач виконує дизасемблювання виконуваного файлу та отримує вихідний код. 2. Програма надає текст з отриманим вихідним кодом для розрахунку ентропії. 3. Програма отримує вхідні дані і оброблює текст для розрахунку ентропії. 4. Програма обчислює ентропію вхідних даних за відповідною формулою. 5. Програма виводить обчислену ентропію на екран.

Продовження таблиці 2.5

1	2
Базовий потік	1.Програма надає текст з отриманим вихідним кодом для розрахунку ентропії. 2.Програма отримує вхідні дані і оброблює текст для розрахунку ентропії. 3.Програма обчислює ентропію вхідних даних за відповідною формулою. 4.Програма виводить обчислену ентропію на екран.
Альтернативні потоки	1.Вихідний код виконуваного файлу не був отриманий після дизасемблювання. Розрахунок ентропії не може бути виконаний. 2. Вхідний файл не знайдено або недоступний для обробки. Програма не може розрахувати ентропію. 3. Під час розрахунку ентропії сталася помилка або програма зупинилась. Розрахунок не може продовжуватись.
Постумови	Користувач успішно отримує обчислену ентропію вхідних даних. Програма може бути закрита або користувач може повторити розрахунок з новими вхідними даними.

Таблиця 2.6 – Специфікація використання «Представлення результатів у вигляді таблиці»

Складова	Опис
1	2
Опис прецеденту	Представлення результатів у вигляді таблиці
Дійові особи	Користувач
Передумови	Результати аналізу або обчислень готові до представлення в таблиці
Потік подій	1. Програма формує таблицю на основі отриманих даних і результатів аналізу та обчислень. 2. Програма відображає таблицю на екрані.
Базовий потік	1. Програма формує таблицю на основі отриманих даних і результатів аналізу та обчислень. 2. Програма відображає таблицю на екрані.
Альтернативні потоки	1. Під час формування таблиці сталася помилка або програма зупинилась. Формування таблиці неможливе.

Продовження таблиці 2.6

1	2
Постумови	Користувач отримує автоматично згенеровану таблицю, яку може використати для подальшого аналізу. Програма може бути закрита або користувач може повторити процес отримання таблиці з новими даними.

Таблиця 2.7 – Специфікація використання «Представлення результатів у вигляді графіка»

Складова	Опис
Опис прецеденту	Представлення результатів у вигляді графіка
Дійові особи	Користувач
Передумови	Результати аналізу або обчислень готові до представлення у вигляді графіка
Потік подій	1. Програма формує таблицю на основі отриманих даних і результатів аналізу та обчислень. 2. Програма відображає графік на екрані.
Базовий потік	1. Програма формує таблицю на основі отриманих даних і результатів аналізу та обчислень. 2. Програма відображає графік на екрані.
Альтернативні потоки	1. Під час формування графіка сталася помилка або програма зупинилась. Формування графіка неможливе.
Постумови	Користувач успішно отримує представлення результатів у вигляді графіка, яку може використати для подальшого аналізу. Програма може бути закрита або користувач може повторити процес побудови графіка з новими даними.

Таблиця 2.8 – Специфікація використання «Порівняння з попередньою версією»

Складова	Опис
1	2
Опис прецеденту	Порівняння з попередньою версією
Дійові особи	Користувач
Передумови	Попередня версія файлу або дані для порівняння існують

Продовження таблиці 2.8

1	2
Потік подій	1. Користувач вибирає файл або дані для порівняння з попередньою версією. 2. Програма виконує порівняння вибраних файлів або даних з попередньою версією. 3. Програма формує таблицю результатів порівняння з попередньою версією. 4. Програма відображає таблицю на екрані. 5. Програма відображає графік на екрані.
Базовий потік	1. Користувач вибирає файл або дані для порівняння з попередньою версією. 2. Програма виконує порівняння вибраних файлів або даних з попередньою версією. 3. Програма формує таблицю результатів порівняння з попередньою версією. 4. Програма відображає таблицю на екрані. 5. Програма відображає графік на екрані.
Альтернативні потоки	1. Користувач не вказує необхідний файл або дані для порівняння. Виведення результатів неможливе. 2. Під час порівняння сталася помилка або недостатньо даних для порівняння.
Постумови	Результати порівняння відображені у вигляді таблиці та графіка на екрані.

Виконаний аналіз вимог [15] допоможе забезпечити розробку потужного та надійного інструменту для виявлення внесених змін в коді виконуваних файлів, що сприятиме підвищенню безпеки програм та захисту від шкідливих змін.

2.2 Розробка архітектури програмного забезпечення для знаходження внесених змін в код виконуваних файлів

Розробка архітектури програмного забезпечення для знаходження внесених змін в код виконуваних файлів є важливим завданням у сфері програмної розробки. При розвитку великих проектів, особливо коли беруть участь кілька розробників, виявлення змін, що були зроблені в коді виконуваних файлів, може бути складним завданням. Цей процес вимагає ефективного і автоматизованого механізму, який

дозволить швидко і точно визначити внесені зміни та їх вплив на програмне забезпечення в цілому. У цьому розділі буде розглянута архітектура програмного забезпечення, яке забезпечує знаходження внесених змін в код виконуваних файлів, включаючи основні компоненти, взаємодію між ними та особливості реалізації.

2.2.1 Розробка діаграми компонентів

Розробка архітектури програмного забезпечення для знаходження внесених змін в код виконуваних файлів буде базуватися на архітектурному шаблоні Model-View-Controller (MVC) [16] . Використання цієї архітектури дозволить досягти поділу відповідальностей та кращого організації коду програми. Розглянемо деталі кожного компонента архітектури:

1. Модель (Model):

— Модель відповідатиме за зберігання та управління даними, пов'язаними з виконуваними файлами та їх змінами.

— Модель може включати структуру, що відображає файлову систему або репозиторій з версіями файлів.

— Вона також міститиме логіку аналізу та порівняння внесених змін.

2. Уявлення (View):

— Уявлення буде відповідати за візуалізацію результатів аналізу та змін в коді виконуваних файлів.

— Воно може представляти графічний інтерфейс користувача, веб-сторінку або інші способи відображення даних.

3. Контролер (Controller):

— Контролер оброблятиме запити, що надходять від користувача або інших компонентів програмного забезпечення.

— Він буде координувати роботу моделі та уявлення, забезпечуючи потік даних та логіку взаємодії між ними.

— Контролер також може ініціювати аналіз виконуваних файлів та передавати результати моделі для подальшого відображення.

Завдяки використанню архітектури MVC, ми забезпечимо:

— Поділ відповідальностей між компонентами, що спростить розробку та підтримку програми.

— Кращу керованість потоку даних завдяки контролеру, який відповідає за обробку різних типів запитів та взаємодію з моделлю та уявленням.

— Гнучкішу архітектуру, яка дозволяє змінювати компоненти програми незалежно один від одного, спрощуючи розширення та модифікацію.

Ця архітектура допоможе забезпечити ефективну розробку програмного забезпечення для виявлення внесених змін в код виконуваних файлів та подальше їх відображення.

Нижче, на рисунку 2.2 представлено діаграму компонентів в загальному вигляді.

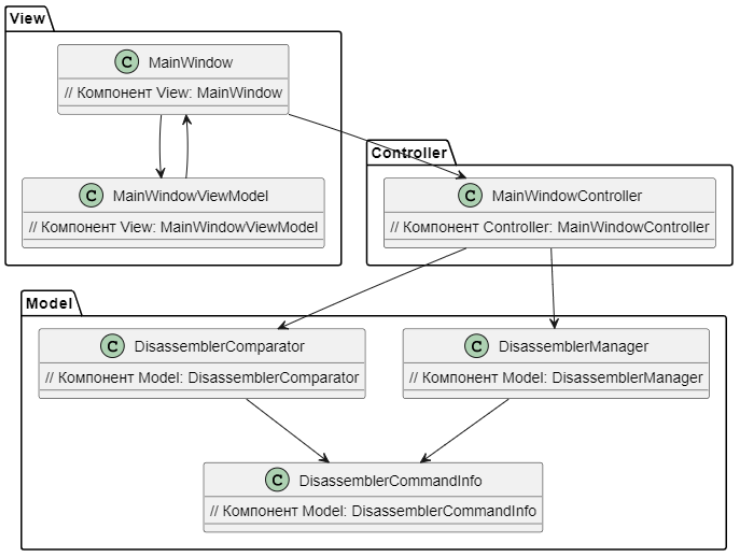

Рисунок 2.2 – Діаграма компонентів програмного забезпечення для знаходження внесених змін в код виконуваних файлів

На цій діаграмі представлено високорівневу структуру архітектури програмного забезпечення з використанням підходу MVC (Model-View-Controller). Ця діаграма надає огляд компонентів та їх взаємозв'язків у програмному забезпеченні.

На діаграмі "MainWindow" виступає як представлення (View), відображаючи дані та взаємодіючи з користувачем. "MainWindowController" виступає як контролер (Controller), він обробляє події та керує взаємодією між представленням та моделлю.

"MainWindowViewModel" виступає як модель (Model) представлення, вона містить дані та бізнес-логіку, яка використовується для оновлення представлення та обробки даних.

Компоненти "DisassemblerCommandInfo", "DisassemblerComparator" та "DisassemblerManager" також належать до моделі (Model) і виконують конкретні завдання, пов'язані з розкомпілюванням та аналізом коду.

Отже, ця діаграма надає огляд компонентів та їх взаємозв'язків в архітектурі програмного забезпечення, демонструючи високорівневу структуру програмного забеззпечення на основі популярного шаблону проектування MVC. На рисунку 2.3 представлено ланцюжок взаємодії компонентів.

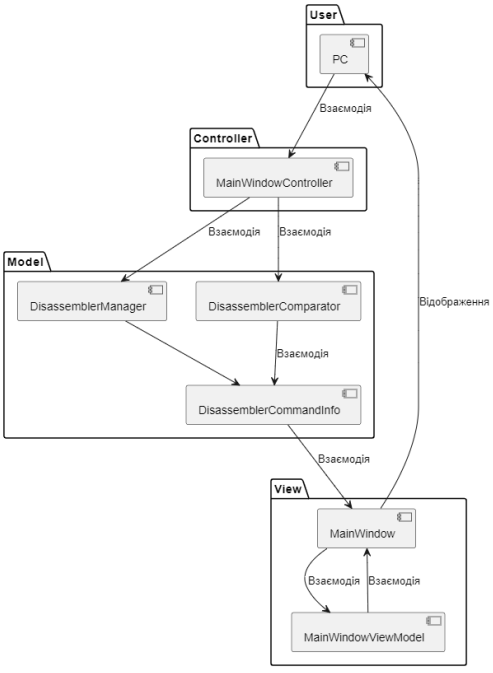

Рисунок 2.3 – Діаграма взаємодії компонентів програмного забезпечення для знаходження внесених змін в код виконуваних файлів

На даній діаграмі показана архітектура програмного забезпечення, що складається з трьох основних компонентів: User (користувач), Model (модель) та Controller (контролер). Компонент User представлений персональним комп'ютером (PC), який використовується для взаємодії з програмним забезпеченням через інтерфейс користувача (UI). Компонент Model містить класи, пов'язані з розбиранням команд (DisassemblerCommandInfo), керуванням розбирачами (DisassemblerManager) та порівнянням розбірок (DisassemblerComparator). Компонент Controller представлений класом MainWindowController, який відповідає за обробку запитів від користувача і взаємодію з іншими компонентами програмного забезпечення.

Для візуалізації компонентів View та Model були використані додаткові пакети. Пакет View містить класи MainWindow та MainWindowViewModel, які відповідають за відображення основного вікна програми та управління даними цього вікна. Пакет Model включає класи DisassemblerCommandInfo, DisassemblerManager та DisassemblerComparator, які представляють логіку розбирання команд, керування розбирачами та порівняння розбірок відповідно.

У діаграмі показані взаємодії між компонентами. MainWindowController взаємодіє з DisassemblerComparator та DisassemblerManager для обробки запитів користувача та керування розбирачами. DisassemblerManager взаємодіє з DisassemblerCommandInfo для отримання інформації про розбирання команд. DisassemblerComparator також взаємодіє з DisassemblerCommandInfo для отримання необхідних даних для порівняння розбірок. MainWindow взаємодіє з MainWindowViewModel для відображення даних у вікні програми. І, нарешті, MainWindowViewModel взаємодіє з MainWindow для обробки взаємодії з користувачем та оновлення вікна.

Загалом, ця діаграма моделю є архітектуру програмного забезпечення, де різні компоненти взаємодіють між собою для обробки запитів користувача, управління розбірачами та відображення даних у вікні програми.

2.2.2 Розробка діаграми розміщення

Для ілюстрації фізичного розгортання програмного забезпечення рекомендовано використовувати діаграму розміщення. Діаграма розміщення, що входить до набору діаграм уніфікованої мови моделювання UML [17], є інструментом для відображення фізичної архітектури програмного забезпечення. Вона дає змогу показати, які програмні та апаратні ресурси використовуються в програмному забезпечені, а також як вони взаємодіють між собою.

Діаграма розміщення зосереджується на фізичних аспектах програмного забезпечення, таких як обладнання, комп'ютери, сервери, додатки, бази даних, мережі та інші фізичні ресурси. Вона надає уявлення про фізичну інфраструктуру, на якій працює програмне забезпечення.

На діаграмі розміщення можна відобразити компоненти програмного забезпечення у вигляді вузлів або контейнерів. Компоненти можуть бути представлені у вигляді блоків або іконок, які відображають фізичні об'єкти, такі як сервери, комп'ютери або мережеві пристрої. З'єднання між компонентами можуть бути показані за допомогою стрілок, що вказують напрямок комунікації або з'єднання.

Діаграма розміщення наведена на рисунку 2.4.

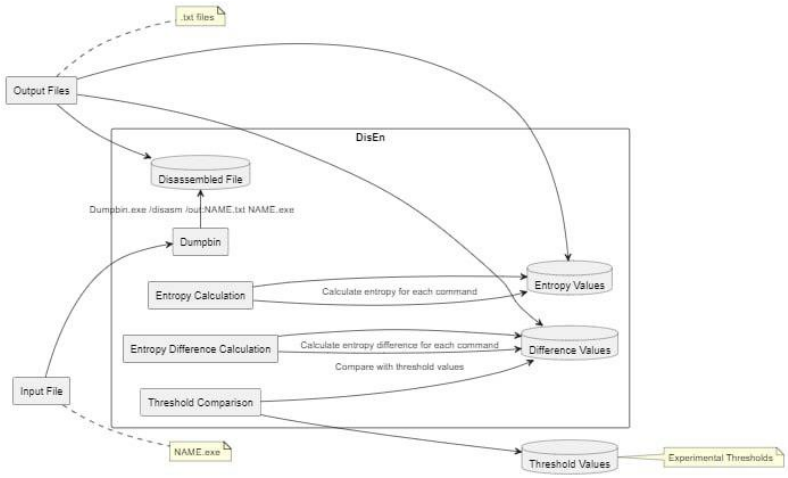

Рисунок 2.4 – Діаграма розміщення програмного забезпечення для знаходження внесених змін в код виконуваних файлів

Наведений вище рисунок побудови діаграми розміщення дозволить розробити проект програмного забезпечення.

2.3 Проект програмного забезпечення

Під час розробки програмного забезпечення для знаходження внесених змін в код виконуваних файлів були використані різноманітні моделювальні інструменти та методології.

2.3.1 Ескізний проект програмного забезпечення

Для формалізації сценаріїв варіантів використання були побудовані діаграми діяльності. Діаграми діяльності є частиною уніфікованої мови моделювання (UML) і дозволяють відображати послідовність дій та умови, які необхідно виконати для їх виконання. Ці діаграми були використані для репрезентації основних сценаріїв варіантів використання.

Окрім цього, була створена інформаційна модель з використанням методології IDEF1X [18]. Ця модель допомогла визначити структуру та зв'язки даних у програмному забезпечені.

В процесі розробки ескізного проекту було розробленно діаграми діяльності та створено проект користувальницького інтерфейсу, який відображає спрощений вигляд програмного забезпечення та його функціональні можливості.

Діаграма діяльності для варіанту використання «Дизасемблювання виконуваного файлу» представлена на рисунку 2.5.

Рисунок 2.5 – Діаграма діяльності для варіанту використання «Дизасемблювання виконуваного файлу»

Ця діаграма діяльності описує послідовність дій користувача та програмного забезпечення під час виконання аналізу виконуваного файлу. Користувач спочатку обирає опцію "Виконати аналіз виконуваного файлу" і завантажує файл, який потрібно проаналізувати. Далі програмне забезпечення перевіряє наявність помилок валідації даних. Якщо помилки виявлено, програмне забезпечення відображає повідомлення про помилку та користувач може її виправити і повторити процес. Якщо помилок немає, програмне забезпечення аналізує код виконуваного файлу та отримує результати аналізу. У кінці програмне забезпечення завершує свою роботу, а користувач отримує результати аналізу.

Діаграма діяльності для варіанту використання «Перегляд вихідного тексту» представлена на рисунку 2.6.

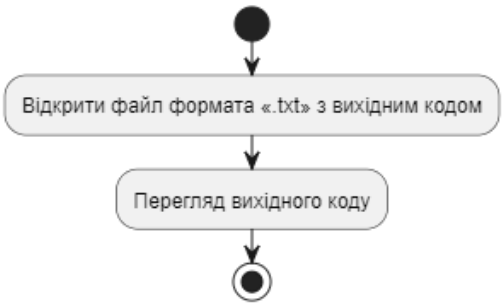

Рисунок 2.6 – Діаграма діяльності для варіанту використання «Перегляд вихідного тексту»

Ця діаграма діяльності відображає послідовність кроків, які користувач виконує для перегляду вихідного коду з файлу формату «.txt». Користувач спочатку відкриває файл формату «.txt» з вихідним кодом. Після цього він може переглядати вихідний код. Коли користувач закінчує перегляд, процес завершується. Діаграма також враховує альтернативний потік, де користувач не має файлу з вихідним кодом або стикається з помилкою під час перегляду.

Діаграму діяльності для варіанту використання «Розрахунок ентропії» представлено на рисунку 2.7.

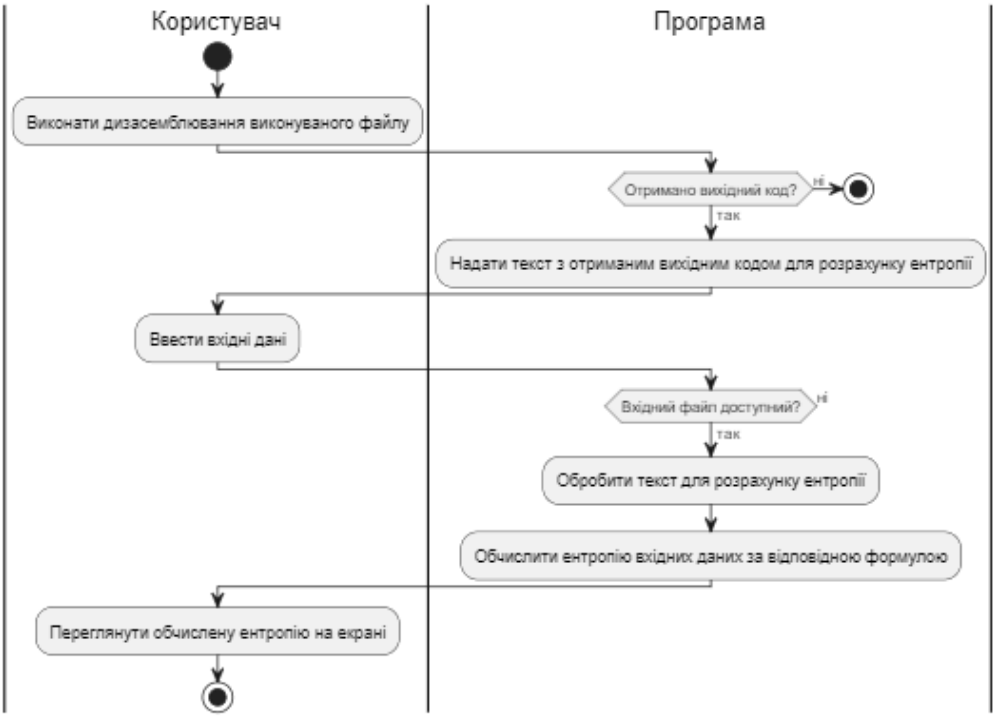

Рисунок 2.7 – Діаграма діяльності для варіанту використання «Розрахунок ентропії»

Наведена діаграма демонструє процес розрахунку ентропії за допомогою програми. Основна діяльність виконується користувачем, який ініціює весь процес. Опис процесу розрахунку ентропії включає наступні кроки:

– Користувач виконує дизасемблювання виконуваного файлу, що призводить до отримання вихідного коду програми.

– Якщо вихідний код був успішно отриманий, програма надає текст з отриманим вихідним кодом для подальшого розрахунку ентропії. Якщо вихідний код недоступний, виконання програми припиняється.

– Користувач вводить вхідні дані, які будуть використовуватись для розрахунку ентропії.

– Програма оброблює вхідні дані, щоб підготувати їх для розрахунку ентропії.

– Програма застосовує відповідну формулу для обчислення ентропії вхідних даних.

– Обчислена ентропія виводиться на екран для перегляду користувачем.

– Якщо вихідний код не був отриманий або вхідний файл недоступний, або сталася помилка під час розрахунку, виконання програми припиняється.

Діаграма відображає послідовність кроків, які потрібно виконати для успішного розрахунку ентропії. Це включає вхідні дані, обробку тексту, обчислення ентропії та виведення результату. У разі невиконання передумов (наявність вихідного коду та доступність вхідного файлу) або виникнення альтернативних ситуацій (відсутність вихідного коду, недоступність вхідного файлу, помилка під час розрахунку) виконання програми припиняється, і користувач повинен вжити відповідних дій для виправлення проблеми.

Діаграму діяльності для варіанту використання «Представлення результатів у вигляді таблиці» представлено на рисунку 2.8

Рисунок 2.8 – Діаграма діяльності для варіанту використання «Представлення результатів у вигляді таблиці»

Ця діаграма діяльності відображає процес представлення результатів у вигляді таблиці для користувача. Вона описує послідовність дій, які відбуваються, починаючи з отримання результатів аналізу або обчислень і закінчуючи відображенням таблиці на екрані.

Спочатку користувач отримує результати аналізу та обчислень. Потім програма перевіряє, чи ці результати готові до представлення. Якщо так, вона переходить до формування таблиці на основі цих даних. Для цього програма використовує отримані дані та результати інших обчислень.

Після формування таблиці програма відображає її на екрані, де користувач може переглянути результати аналізу. Це дозволяє користувачу легко спостерігати і аналізувати дані у структурованому форматі.

У випадку, якщо результати аналізу або обчислень не готові, програма повідомляє користувача про це, що дозволяє йому знати, що потрібно зробити, щоб отримати доступ до результатів.

Ця діаграма діяльності надає загальний огляд процесу представлення результатів у вигляді таблиці та показує послідовність дій, які виконуються для забезпечення ефективного та зручного способу відображення результатів аналізу користувачеві.

Діаграму діяльності для варіанту використання «Представлення результатів у вигляді графіка» представлено на рисунку 2.9

Рисунок 2.9 – Діаграма діяльності для варіанту використання «Представлення результатів у вигляді графіка»

Ця діаграма діяльності відображає процес представлення результатів у вигляді графіка для користувача. Вона описує послідовність дій, які відбуваються, починаючи з отримання результатів аналізу або обчислень і закінчуючи відображенням графіка на екрані.

Спочатку користувач отримує результати аналізу та обчислень. Потім програма перевіряє, чи ці результати готові до представлення. Якщо так, вона переходить до формування таблиці на основі цих даних. Для цього програма використовує отримані дані та результати інших обчислень.

Після формування таблиці програма перевіряє готовність до побудови графіка. Якщо дані підходять для графіка, програма створює графік з використанням цих даних. Вона враховує розміщення стовпців, масштабування осей та інші налаштування.

Після побудови графіка програма відображає її на екрані для користувача. Це дозволяє користувачу легко спостерігати і аналізувати дані, представлені у формі графіка, що надає візуальне уявлення про розподіл даних.

У випадку, якщо результати аналізу або обчислень не готові для побудови графіка або сталася помилка, програма повідомляє користувача про це, що дозволяє йому знати, що потрібно зробити, щоб отримати доступ до результатів у формі графіка.

Ця діаграма діяльності надає загальний огляд процесу представлення результатів у вигляді графіка та показує послідовність дій, які виконуються для забезпечення ефективного та зручного способу відображення результатів аналізу у вигляді графіка користувачеві.

Діаграму діяльності для варіанту використання «Порівняння з попередньою версією» представлено на рисунку 2.10.

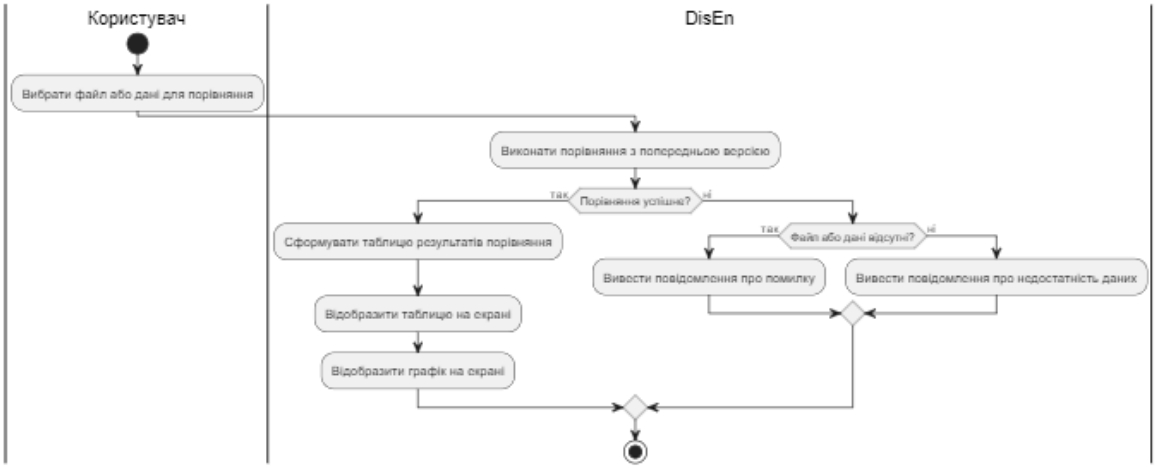

Рисунок 2.10 – Діаграма діяльності для варіанту використання «Порівняння з попередньою версією»

Наступним етапом розробки ескізного проекту програмного забезпечення для знаходження внесених змін в код виконуваних файлів являється розробка інтерфейсу. Нижче будуть представлені ескізи інтерфейсу.

Інтерфейс користувача є важливою складовою програмного забезпечення. Від нього залежить ефективність роботи користувача та точність управління

програмою. За допомогою прототипу можна перевірити сприйняття програмне забезпечення користувачем та основну логіку її функціонування. Використання паперових прототипів має свої переваги, такі як простота модифікації на основі результатів тестування та можливість безболісного залучення представників цільової аудиторії. На паперовому етапі прототип може пройти кілька змін і, відповідно, бути представлений у різних варіантах. Рисунки 2.11 – 2.14 демонструють вікна програми.

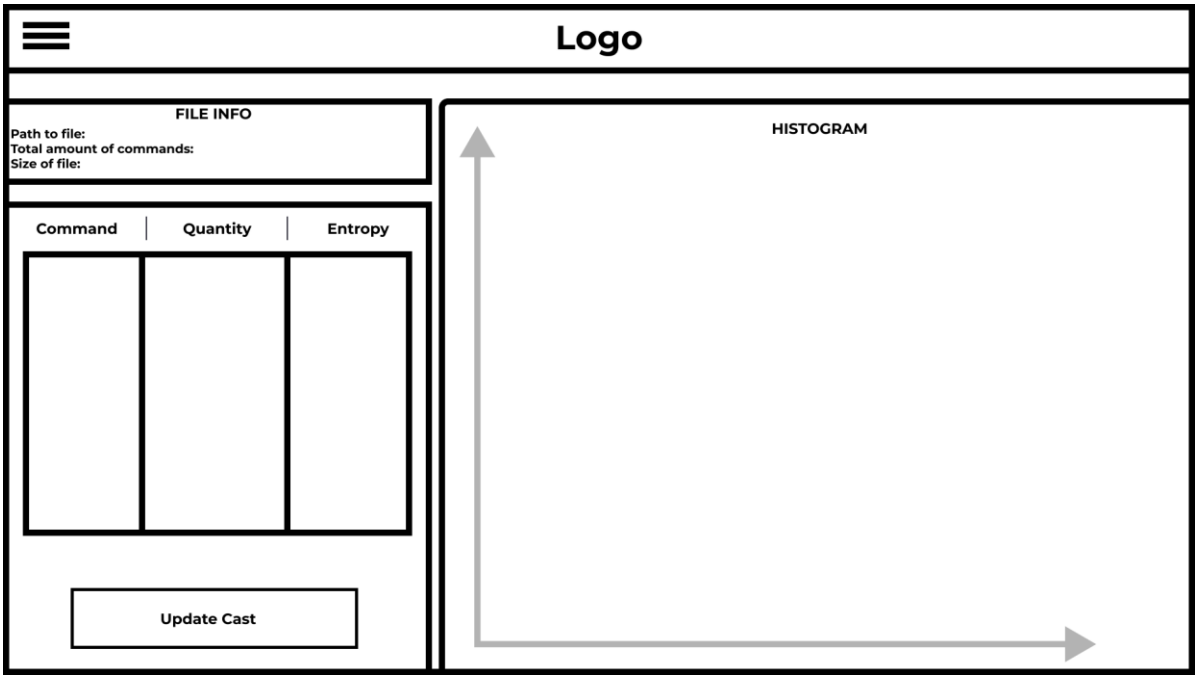

Рисунок 2.11 – Прототип головного вікна програми

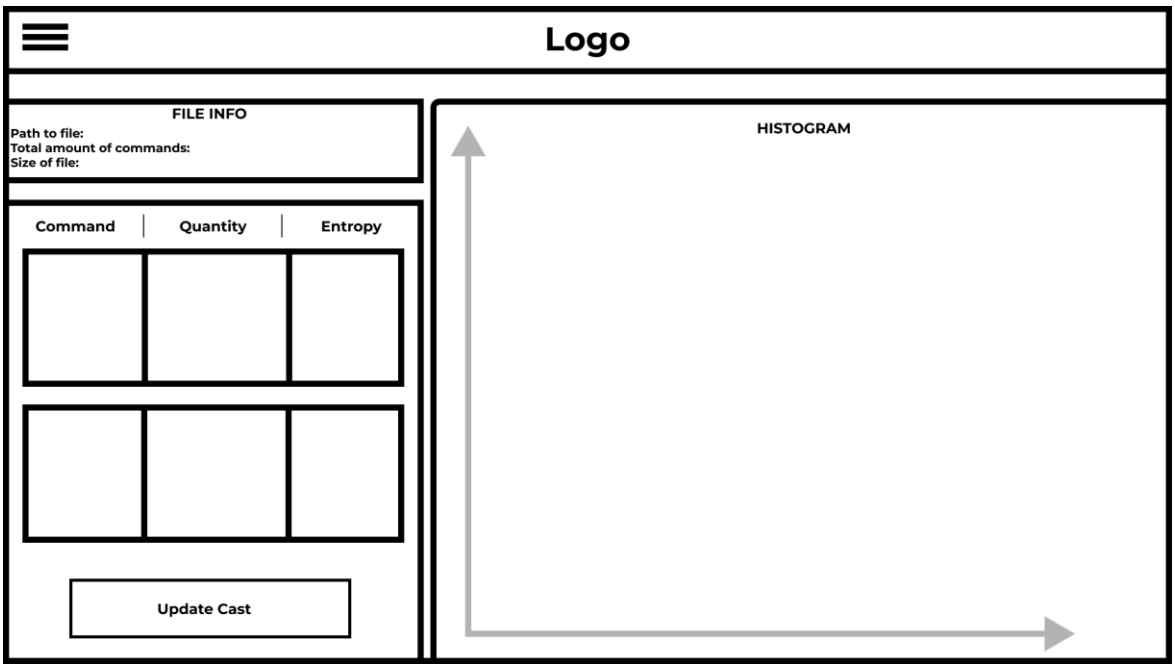

Рисунок 2.12 – Прототип головного вікна програми під час демонстрації результатів досліджень двох файлів

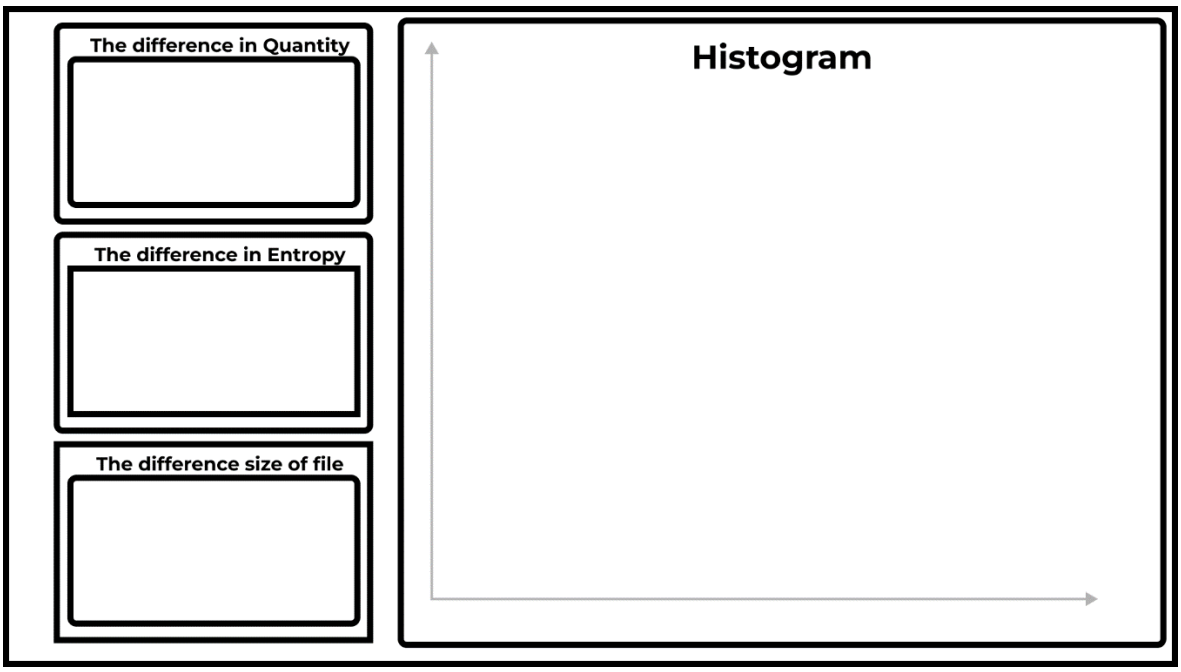

Рисунок 2.13 – Прототип вікна детальної інформації про дослідження двох файлів

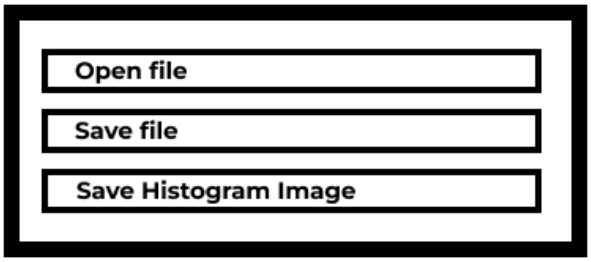

Рисунок 2.14 – Прототип вінка підрозділу «File» головного вікна випадаючого списку меню

Далі можемо після виконання ескізного проекту далі можемо переходити до розробки технічного проекту програмного забезпечення.

2.3.2 Технічний проект програмного забезпечення

Після ретельного аналізу ескізного проекту було розроблено технічний проект програмного забезпечення під назвою "DisEn". Для досягнення цієї мети були побудовані статична та динамічна моделі, які ілюструють роботу програмне забезпечення з різних боків.

Статична модель програмного забезпечення представлена у вигляді діаграми класів, яка відображає взаємозв'язки між класами та структуру програми. Вона надає загальний огляд архітектури розроблюваного сервісу. З іншого боку, динамічна модель програмне забезпечення представлена у вигляді діаграм послідовності, яка демонструє взаємодію компонентів програмне забезпечення та послідовність виконання операцій.

Приведена статична модель розроблюваного сервісу на діаграмі класів показує його складові частини, включаючи класи, атрибути та методи, а також зв'язки між ними. Вона допомагає зрозуміти структуру програми та взаємозв'язки між її компонентами.

Рисунок 2.15 Діаграма класів програмного забезпечення для знаходження внесених змін в код виконуваних файлів

Специфікації класів програмного забезпечення «DisEn» для знаходження внесених змін в код виконуваних файлів приведено нижче:

Клас DisassemblerManager

Назва: DisassemblerManager

Короткий опис: Клас, відповідальний за управління дизасемблером та обробку виконуваних файлів.

Атрибути:

— _disassemblerComparator: Об'єкт класу DisassemblerComparator для порівняння даних дизасемблера.

Методи:

— Disassemble(filename: string): Метод, який дизасемблює виконуваний файл з вказаним ім'ям.

— GetCurrentDisassembler(): Метод, який повертає поточний об'єкт дизасемблера.

— GetSavedDisassembler(): Метод, який повертає збережений об'єкт дизасемблера.

— SaveCurrentDisassemblerFile(): Метод, який зберігає поточний файл дизасемблера.

— CompareData(disassembler1: Disassembler, disassembler2: Disassembler): Метод, який порівнює дані між двома об'єктами дизасемблера.

Клас DisassemblerComparator

Назва: DisassemblerComparator

Короткий опис: Клас, що виконує порівняння результатів дизасемблювання між двома об'єктами класу Disassembler.

Атрибути:

— _firstDisassembler: Об'єкт класу Disassembler, що містить перший дизасемблер.

— _secondDisassembler: Об'єкт класу Disassembler, що містить другий дизасемблер.

— _totalEntropyDelta: Дійсне число, що представляє різницю загальної ентропії між двома дизасемблерами.

— _totalInstructionCounterDelta: Ціле число, що представляє різницю загальної кількості інструкцій між двома дизасемблерами.

— _fileSizeDelta: Дійсне число, що представляє різницю розміру файлу між двома дизасемблерами.

— _disassemblerCommandInfoDelta: Список об'єктів класу DisassemblerCommandInfo, що містить різницю в інформації про команди між двома дизасемблерами.

Методи:

— GetFirstDisassembler(): Метод, що повертає перший дизасемблер.

— GetSecondDisassembler(): Метод, що повертає другий дизасемблер.

— GetTotalEntropyDelta(): Метод, що повертає різницю загальної ентропії.

— GetTotalInstructionCounterDelta(): Метод, що повертає різницю загальної кількості інструкцій.

— GetFileSizeDelta(): Метод, що повертає різницю розміру файлу.

— GetDisassemblerCommandInfoDelta(): Метод, що повертає список об'єктів DisassemblerCommandInfo з різницею в інформації про команди.

— CompareData(firstDisassembler: Disassembler, secondDisassembler: Disassembler): Метод, що порівнює дані між двома дизасемблерами і обчислює відповідні різниці. Повертає true, якщо дані ідентичні, і false в іншому випадку.

Клас DisassemblerCommandInfo

Назва: DisassemblerCommandInfo

Короткий опис: Клас, що представляє інформацію про команду дизасемблера.

Атрибути:

— Name: Рядок, що містить назву команди.

— Count: Ціле число, що вказує кількість зустрічей команди.

— Entropy: Дійсне число, що представляє ентропію команди.

Клас DisassemblerCommandInfo

Назва: Disassembler

Короткий опис: Клас, що відповідає за дизасемблювання файлів.

Атрибути:

— _fileName: Рядок, що містить ім'я файлу.

— _parsedFileCode: Рядок, що містить розібраний код файлу.

— _filePath: Рядок, що містить шлях до дизасембльованого файлу.

— _totalInstructionCounter: Ціле число, що показує загальну кількість інструкцій.

— _totalEntropyValue: Дійсне число, що представляє загальну ентропію файлу.

— _fileSize: Дійсне число, що вказує розмір файлу в байтах.

— _instructionFilterHashSet: Об'єкт класу HashSet<string>, що містить фільтр інструкцій.

— _instructionsDict: Об'єкт класу Dictionary<string, Int32>, що містить словник інструкцій та їх кількості.

— _disassemblerCommandsInfo: Список об'єктів класу DisassemblerCommandInfo, що містить інформацію про команди.

Методи:

— Disassembler: Конструктор класу, ініціалізує контейнери та встановлює _totalInstructionCounter в нуль.

— Equals: Перевизначений метод з Object, порівнює дані двох об'єктів типу Disassembler.

— GetHashCode: Перевизначений метод з Object, повертає хеш-код об'єкта.

— SetInstructionFilterHashSet(InstructionFilterHashSet: HashSet<string>): Метод, що встановлює фільтр інструкцій.

— GetFileName(): Метод, що повертає ім'я файлу.

— GetFilePath(): Метод, що повертає шлях до дизасембльованого файлу.

— GetInstructionFilterHashSet(): Метод, що повертає фільтр інструкцій.

— GetTotalInstructionCounter(): Метод, що повертає загальну кількість інструкцій.

— GetFileSize(): Метод, що повертає розмір файлу.

— GetFileTotalEntropyValue(): Метод, що повертає загальну ентропію файлу.

— GetDisassemblerCommandsInfo(): Метод, що повертає список об'єктів DisassemblerCommandInfo.

— Serialize(path: string, disassembler: Disassembler): Статичний метод, що серіалізує об'єкт Disassembler в файл за заданим шляхом.

— Deserialize(path: string): Статичний метод, що десеріалізує об'єкт Disassembler з файлу за заданим шляхом.

— DisassembleFile(filePath: string): Метод, що дизасемблює файл за заданим шляхом.

— ExportDisassembledCodeToTxtFile(fileExportPath: string): Метод, що експортує розібраний код в текстовий файл за заданим шляхом.

— ParseDisassembledTxtFile(filePath: string): Приватний метод, що розбирає текстовий файл з дизасембльованим кодом.

— DisassembleExeToTxtFile(filePath: string, fileSavePath: string): Приватний метод, що дизасемблює файл і зберігає результат у текстовому форматі.

— FormCommandsInfo(): Приватний метод, що формує інформацію про команди.

— CalculateFileTotalEntropyValue(): Метод, що обчислює загальну ентропію файлу.

— ParseDisassemble(dissFileLines: string[]): Приватний метод, що розбирає дизасембльований код інструкцій.

— EntropyCalculation(elementsCount: double, allElementsCount: double): Приватний метод, що обчислює ентропію.

— CompareDisInfo(disComInfo: List<DisassemblerCommandInfo>): Приватний метод, що порівнює інформацію про команди.

Клас DisassemblerCommandInfo

Назва: ByteConverter

Короткий опис: Клас, що містить методи для конвертації байтів в мегабайти.

Атрибути: відсутні.

Методи:

— ConvertByteToMegaByte(byteToConvert: double): Статичний метод, що конвертує вхідне значення байтів у мегабайти.

— ConvertByteToMegaByteToString(byteToConvert: double): Статичний метод, що конвертує вхідне значення байтів у мегабайти і повертає рядок з результатом.

У наступному етапі розробки технічного проекту програмного забезпечення, ми зосереджуємося на побудові динамічної моделі для розроблюваного програмного забезпечення. Використовуючи раніше розроблені моделі, зокрема

модель варіантів використання та діаграму класів, ми побудували динамічну модель. На рисунках 2.17 – 2.22 представлені діаграми послідовності для основних варіантів використання, які втілюють цю модель.

Діаграма послідовності для прецеденту «Дизасемблювання виконуваного файлу» можна побачити на рисунку 2.16.

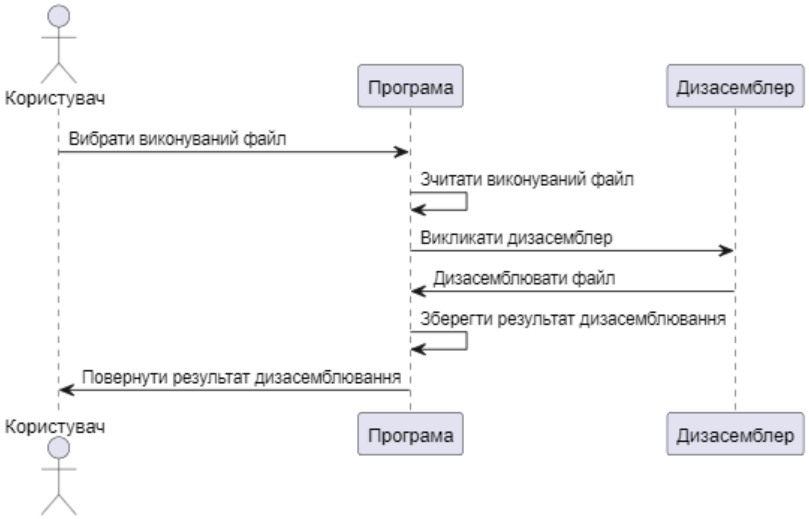

Рисунок 2.16 – Діаграма послідовності для прецеденту « Дизасемблювання виконуваного файлу»

У цій діаграмі послідовності показано взаємодію між користувачем (User) та програмою для прецеденту "Дизасемблювання виконуваного файлу". Користувач вибирає виконуваний файл, який передається програмі. Програма зчитує виконуваний файл і викликає дизасемблер для його дизасемблювання. Дизасемблер дизасемблює файл і передає результат програмі. Програма зберігає результат дизасемблювання і повертає його користувачу.

Діаграма послідовності для прецеденту «Перегляд вихідного тексту» представлена на рисунку 2.17

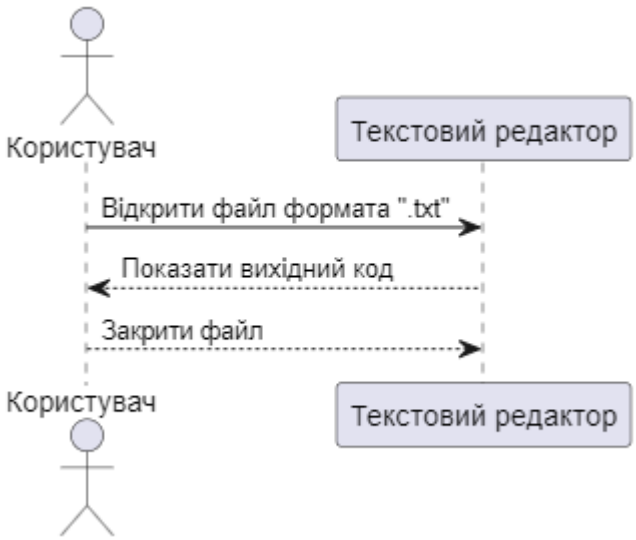

Рисунок 2.17 – Діаграма послідовності для прецеденту « Перегляд вихідного тексту»

У цій діаграмі послідовності показано взаємодію між користувачем та текстовим редактором під час перегляду вихідного коду. Діаграма починається зі стартової точки, де користувач відкриває файл формата ".txt" у текстовому редакторі. Потім відбувається взаємодія між користувачем та редактором, де останній показує вихідний код користувачу. Після цього користувач може взаємодіяти з вихідним кодом, таким як редагування або збереження змін, але ці дії не відображені на діаграмі для спрощення. Нарешті, користувач закриває файл, і взаємодія між ним і текстовим редактором завершується. Діаграма показує послідовність подій та взаємодію між основними учасниками в процесі перегляду вихідного тексту.

Діаграма послідовності для прецеденту «Розрахунок ентропії» представлена на рисунку 2.18

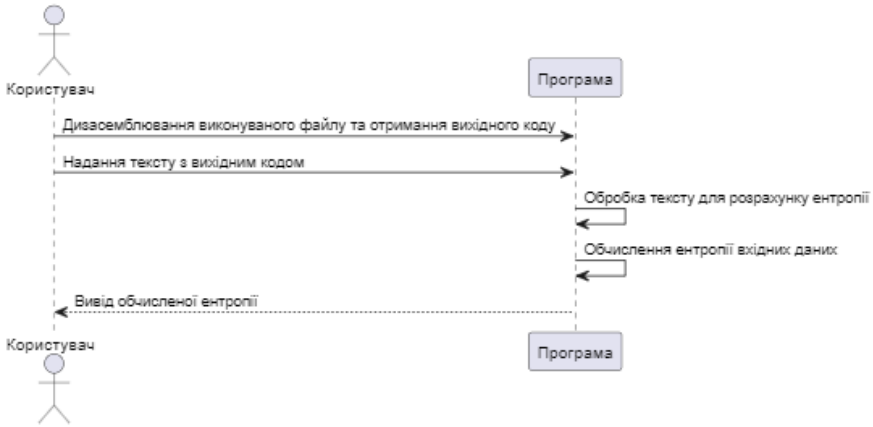

Рисунок 2.18 – Діаграма послідовності для прецеденту « Розрахунок ентропії»

У цій діаграмі послідовності показано взаємодію між користувачем і програмою під час розрахунку ентропії. Діаграма починається з дії користувача – дизасемблювання виконуваного файлу та отримання вихідного коду. Потім користувач передає цей вихідний код програмі для розрахунку ентропії. Програма оброблює текст, обчислює ентропію вхідних даних за відповідною формулою і виводить обчислену ентропію на екран. Діаграма показує послідовність подій та взаємодію між користувачем та програмою в процесі розрахунку ентропії.

Діаграма послідовності для прецеденту «Представлення результатів у вигляді таблиці» представлена на рисунку 2.19

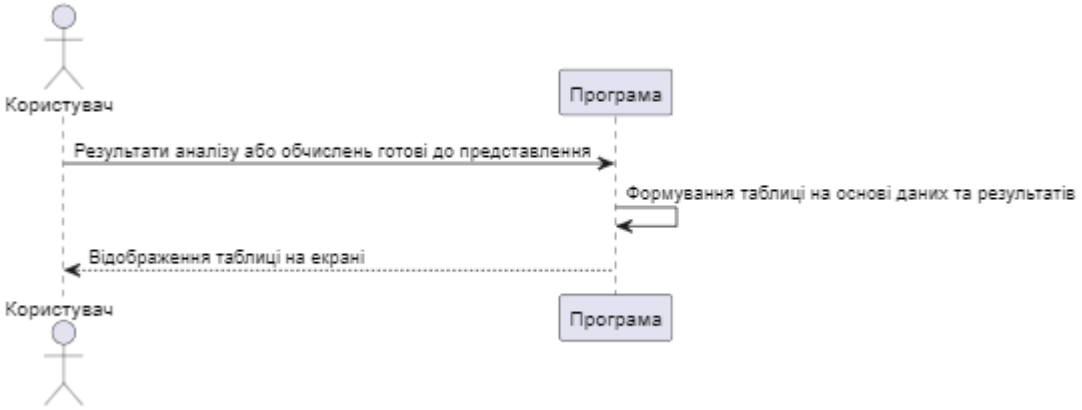

Рисунок 2.19 – Діаграма послідовності для прецеденту «Представлення результатів у вигляді таблиці»

У цій діаграмі послідовності показана взаємодія між користувачем і програмою під час представлення результатів у вигляді таблиці. Користувач передає програмі результати аналізу або обчислень, які готові для представлення. Програма формує таблицю на основі цих даних та результатів і відображає її на екрані для користувача. Діаграма показує послідовність подій і взаємодію між користувачем та програмою в процесі представлення результатів у вигляді таблиці.

Діаграма послідовності для прецеденту «Представлення результатів у вигляді графіка» представлена на рисунку 2.20

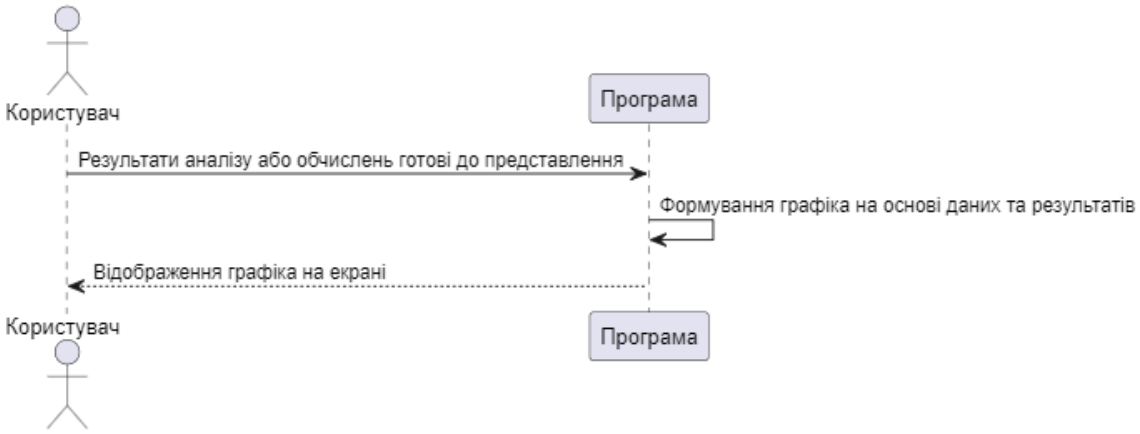

Рисунок 2.20 – Діаграма послідовності для прецеденту «Представлення результатів у вигляді графіка»

У цій діаграмі послідовності показана взаємодія між користувачем і програмою під час представлення результатів у вигляді графіка. Користувач передає програмі результати аналізу або обчислень, які готові для представлення. Програма формує таблицю на основі цих даних та результатів і відображає графік на екрані для користувача. Діаграма показує послідовність подій і взаємодію між користувачем та програмою в процесі представлення результатів у вигляді графіка.

Діаграма послідовності для прецеденту «Порівняння з попередньою версією» представлена на рисунку 2.21

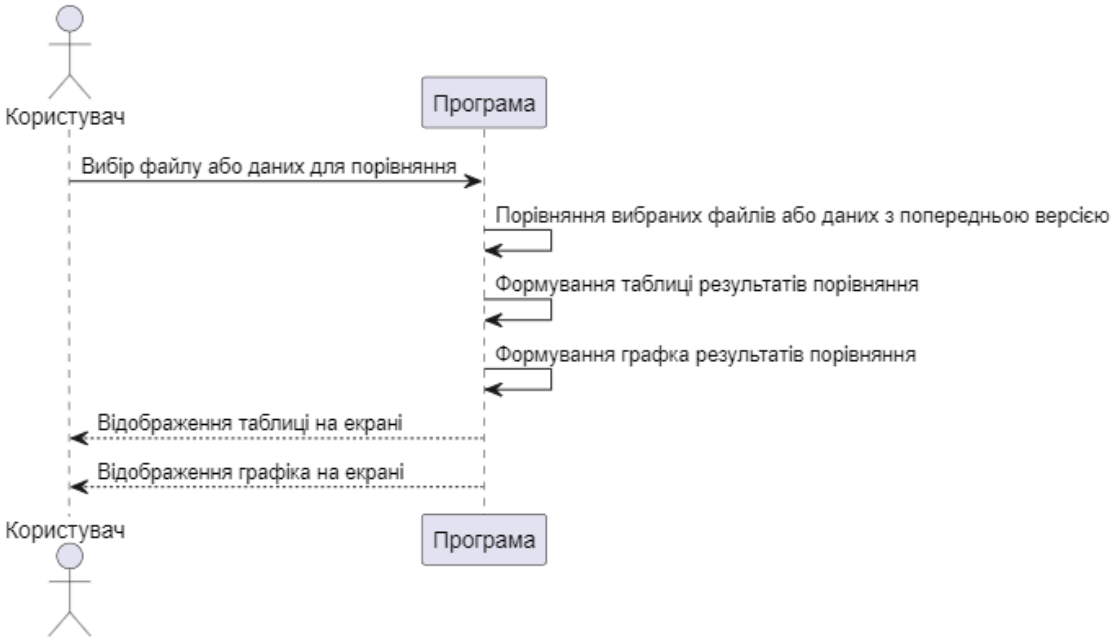

Рисунок 2.21 – Діаграма послідовності для прецеденту «Порівняння з попередньою версією»

У цій діаграмі послідовності показана взаємодія між користувачем і програмою під час порівняння з попередньою версією. Користувач вибирає файл або дані для порівняння, а програма виконує порівняння вибраних файлів або даних з попередньою версією. Потім програма формує таблицю результатів порівняння і відображає її на екрані для користувача. Також програма відображає графік на екрані. Діаграма показує послідовність подій і взаємодію між користувачем та програмою в процесі порівняння з попередньою версією.

2.3.3 Робочий проект програмного забезпечення для знаходження внесених змін в код виконуваних файлів

В якості середовища розробки було обрано Visual Studio. Це повнофункціональне інтегроване середовище розробки (IDE), розроблене компанією Microsoft. Воно надає розробникам зручну та потужну платформу для створення програмного коду. Visual Studio підтримує широкий спектр мов програмування, включаючи C#, C++, Java, Python та інші. З його допомогою можна створювати різноманітні типи проектів, від консольних програм до веб-додатків і мобільних додатків. Крім того, Visual Studio надає багато інструментів для редагування, налагодження та тестування коду, а також підтримку для керування версіями і спільної роботи над проектами. Всі ці можливості роблять Visual Studio незамінним інструментом для професійних розробників програмного забезпечення.

Як графічна підсистема була обрана Windows Presentation Foundation (WPF) [19]. Це графічна підсистема в рамках фреймворку Microsoft .NET, яка надає єдину платформу для створення візуально привабливих та інтерактивних інтерфейсів користувача для десктоп-додатків під Windows. Вона пропонує багатий набір елементів керування, опцій компонування та можливостей прив'язки даних для створення висококастомізованих та реактивних додатків.WPF використовує мову розмітки XAML (eXtensible Application Markup Language), що дозволяє розробникам визначати інтерфейс користувача та логіку додатка в декларативному та відокремленому від коду способі. Ця розділеність полегшує співпрацю між дизайнерами та розробниками.За допомогою WPF розробники можуть створювати візуально захоплюючі додатки, використовуючи потужні функції, такі як векторна графіка, анімація, підтримка мультимедіа та стилі/шаблони. Він також підтримує прив'язку даних, що дозволяє легко синхронізувати елементи інтерфейсу з джерелами даних.Додатки WPF є незалежними від розширення та можуть адаптуватися до різних розмірів екрану та налаштувань DPI, забезпечуючи однорідний досвід користувача на різних пристроях. Крім того, WPF надає

підтримку 3D-графіки, що дозволяє розробникам створювати захоплюючі та інтерактивні 3D-візуалізації всередині своїх додатків. Загалом, WPF – це універсальний фреймворк для створення сучасних та привабливих десктоп-додатків з багатим інтерфейсом користувача, передовими графічними можливостями та безшовною інтеграцією даних та мультимедіа.

Як дізассемблер був обран DumpBin. DumpBin є утилітою командного рядка, яка входить до складу пакету розробки програмного забезпечення Microsoft Visual Studio. Ця утиліта дозволяє отримати детальну інформацію про бінарні файли, такі як виконувані файли (EXE), бібліотеки DLL та об'єктні файли (OBJ). За допомогою DumpBin можна отримати таку інформацію, як таблиці імпорту та експорту, список функцій, символи, залежності від зовнішніх бібліотек, ресурси, таблиці розташування та багато іншого. Вона допомагає розробникам аналізувати та розуміти структуру та характеристики бінарних файлів, що є корисним для налагодження, профілювання та відлагодження проблем в програмах. DumpBin є потужним інструментом для аналізу бінарних файлів та дозволяє розробникам отримати глибоке розуміння структури та внутрішніх характеристик файлів, що сприяє вдосконаленню процесу розробки та діагностиці проблем у програмах, розроблених на платформі Windows.

C# як мова програмування було обранно C# [20]. C# (C Sharp) є сучасною об'єктно-орієнтованою мовою програмування, розробленою компанією Microsoft. Вона входить до складу платформи розробки Microsoft .NET і є однією з основних мов програмування для створення додатків на платформі .NET. C# поєднує в собі синтаксичні риси мов C++ і Java, пропонуючи простоту використання та потужність об'єктно-орієнтованого програмування. Вона підтримує багато функцій, таких як успадкування, поліморфізм, інкапсуляція, обробка винятків, делегати, події, LINQ (Language-Integrated Query) і багато іншого. C# є потужним і гнучким інструментом для розробки програмного забезпечення на платформі .NET і дозволяє розробникам створювати широкий спектр додатків з високою продуктивністю та функціональністю.

Тестування програмного забезпечення «DisEn» для знаходження внесених змін в код виконуваних файлів

У ході проведення тестування програмного забезпечення «DisEn» для знаходження внесених змін в код виконуваних файлів, було протестовано функціонал "Виконати аналіз виконуваного файлу".

Вхідними даними для цього тесту є виконуваний файл, який підлягає аналізу. Користувач в програмному забезпеченні «DisEn» знаходить відповідну опцію або функціонал, який дозволяє виконати аналіз виконуваного файлу. Після цього, вказується шлях до виконуваного файлу, що підлягає аналізу.

Очікуваним результатом тесту є успішне виконання аналізу виконуваного файлу програмним забезпеченням «DisEn». Після завершення аналізу, програмне забезпечення відображає результати аналізу.

Таблиця 2.9 – Перелік тестування основних функціональних можливостей

№	Дія	Очікуваний результат	Результат перевірки
1	Дизасемблювання виконуваного файлу	Отримаємо файл формата .txt з вихідним кодом виконуваного файлу	Програма успішно дизасемблювала виконуваний файл, вихідний код виконуваного файлу отримано
2	Перегляд вихідного тексту	Відображення коду у файлі формату .txt	Програма успішно відобразила текст вихідного коду
3	Розрахунок ентропії	Числове значення ентропії для кожної команди з дизасомбльованого виконуваного файлу	Програма успішно розрахувала ентропію для кожної команди з дизасомбльованого виконуваного файлу

Продовження таблиці 2.9

4	Представлення результатів у вигляді таблиці	Таблиця з даними яка містить наступні поля: команди, кількість команд, ентропія команд	Програма успішно представила результати у вигляді таблиці. Строками таблиці є інформація про кожну команду
5	Представлення результатів у вигляді графіка	Графік якій відображає данні про аналіз файлу	Програма успішно представила результати у вигляді графіка
6	Порівняння з попередньою версією	Виявлення різниць між двома версіями виконуваного файлу, та відображення інформації про різницю у вигляді таблиці та графіку	Програма успішно порівняла виконуваний файл з попередньою версією та вивела інформацію про різницю у вигляді таблиці та графіку

Після аналізу "Переліку тестування основних функціональних можливостей" було виявлено, що всі основні функції працюють, їх функціональність відповідає вимогам і задовольняє потреби користувачів, демонструючи високу ефективність та надійність.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ «DisEn» ДЛЯ ЗНАХОДЖЕННЯ ВНЕСЕНИХ ЗМІН В КОД ВИКОНУВАНИХ ФАЙЛІВ

В результаті кваліфікаційної роботи було успішно розроблено програмне забезпечення «DisEn», яке призначене для знаходження внесених змін у код виконуваних файлів.

Це програмне забезпечення повністю задовольняє всі вимоги до функціональних характеристик, організації вхідних та вихідних даних, а також інші вимоги, що були визначені у постановці задачі.

Розроблене програмне забезпечення «DisEn» для знаходження внесених змін в код виконуваних файлів включає реалізацію наступних основних функцій:

– Дизасемблювання виконуваних файлів: "DisEn" забезпечує можливість дизасемблювання виконуваних файлів на вихідний код асемблера. Це дозволяє аналізувати та зрозуміти структуру та логіку програми.

– Перегляд вихідного тексту: Програмне забезпечення надає можливість перегляду дизасембльованого вихідного тексту в зручному форматі. Це дозволяє користувачам аналізувати код та здійснювати необхідні зміни.

– Розрахунок ентропії: «DisEn» проводить розрахунок ентропії виконуваного файлу. Ентропія вказує на ступінь невизначеності в коді та може бути корисною для виявлення внесених змін у код виконуваних файлі, а також складних алгоритмів або вразливостей.

– Представлення результатів у вигляді таблиці: Програма надає можливість представляти результати аналізу в зручному табличному форматі. Це дозволяє користувачам оглядати, фільтрувати та сортувати дані за різними параметрами.

– Представлення результатів у вигляді графіка: Програмне забезпечення також забезпечує візуальне представлення результатів аналізу у вигляді графіка. Це дозволяє швидко оцінити розподіл даних та зробити висновки щодо їх характеристик.

– Порівняння з попередньою версією: Програма дозволяє порівнювати дві версії виконуваного файлу та виявляти внесені зміни. Це корисна функція для виявлення різниці між версіями програми та аналізу їх впливу на функціональність та безпеку.

Також під час розробки програмного забезпечення для знаходження внесених змін у код виконуваних файлів були розроблені додатки такі як:

– Додаток А – Технічне завдання на розробку програмного забезпечення для знаходження внесених змін в код виконуваних файлів. Додаток містить опис технічних вимог та вимог до функціональності програмного забезпечення.

– Додаток Б – Вихідні тексти програмних модулів. Додаток містить реалізацію програмних модулів, які складають програмне забезпечення для знаходження внесених змін в код виконуваних файлів.

– Додаток В – Опис програмного забезпечення. Додаток містить опис функціональності, архітектури та основних компонентів програмного забезпечення для знаходження внесених змін в код виконуваних файлів.

– Додаток Г – Інструкція користувача програмного забезпечення для знаходження внесених змін в код виконуваних файлів. Додаток містить детальну інформацію про використання програми, включаючи установку, налаштування та використання програми.

– Додаток Д – Програма та методика випробувань програмного забезпечення. Додаток містить опис процесу випробування програмного забезпечення, включаючи методики випробування, набори даних для випробування та очікувані результати.

Під час розробки програмного забезпечення було проведене його випробування та випробування, згідно з програмою та методикою, які наведені в Додатку Д. Ці випробування були спрямовані на перевірку функціональності, точності та працездатності програмного продукту. Результати випробування підтвердили повну працездатність програмного забезпечення "DisEn" і його відповідність вимогам.

Таким чином, завершення випробування доводить, що програмне забезпечення "DisEn" готове до використання і може бути успішно впроваджене для виявлення внесених змін в код виконуваних файлів.

На рисунках 3.1 – 3.7 продемонстровано випробування програмного забезпечення «DisEn».

Після запуску програмного забезпечення «DisEn», «DisEn» видає головну сторінку програмного забезпечення, яке представлено на рисунку 3.1.

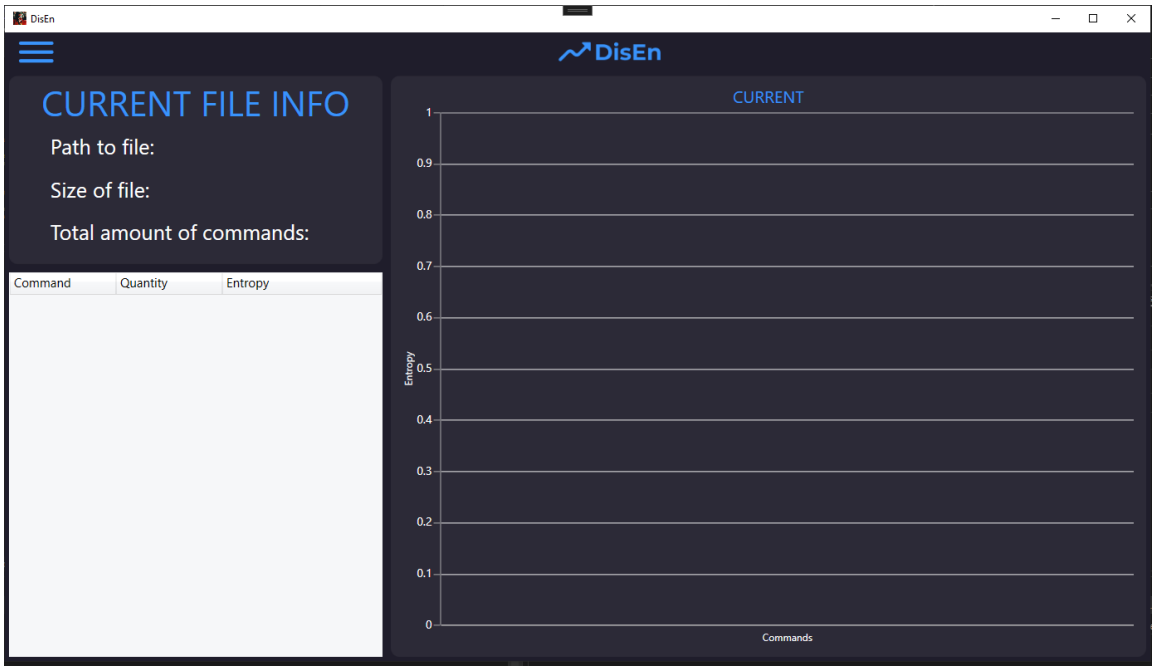

Рисунок 3.1 – «DisEn» після запуску виводить головну сторінку програми.

При наведенні на випадаюче меню ми отримує доступ к переліку функцій програмного забезпечення, яке представлено на рисунку 3.2.

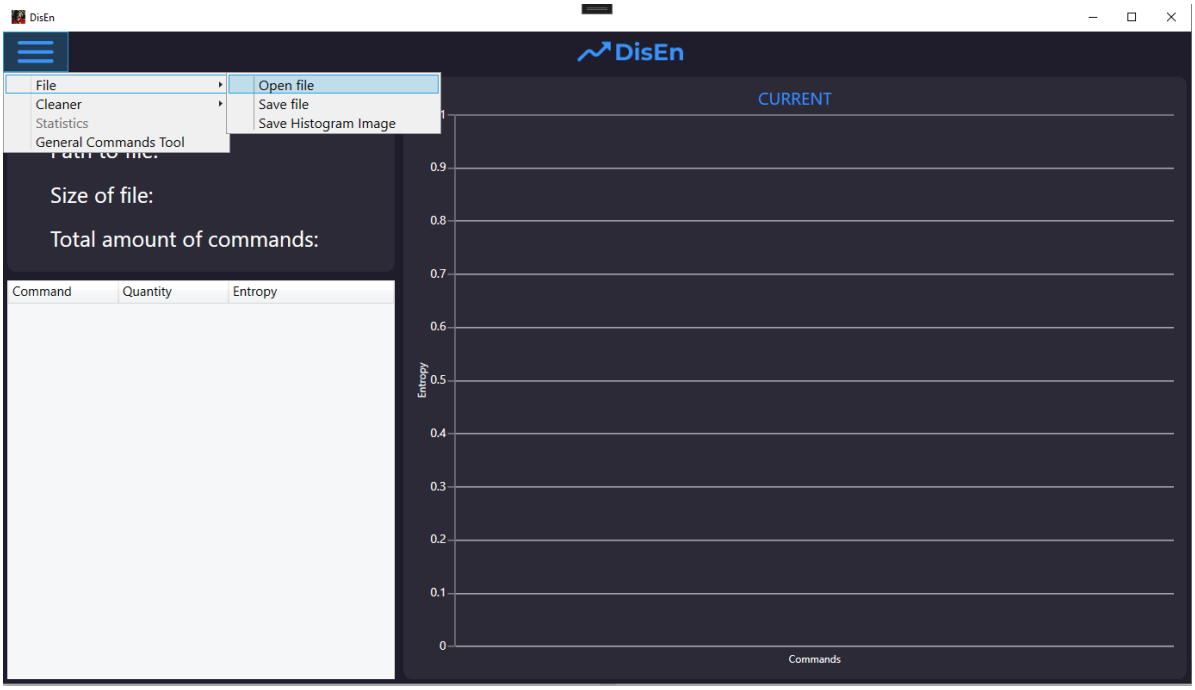

Рисунок 3.2 – Наведенням на випадаюче меню отримуємо доступ до функцій програмного забезпечення.

Після вибору виконуваного файла, «DisEn» починає процес дизасемблювання, та підрахунку ентропії, в випадку знаходження попередньої версії виконуваного файла проводиться порівняння, яке представлено на рисунку 3.3.

Рисунок 3.3 – Після вибору файлу «DisEn» проводить дизасемблювання та підрахунок ентропії. В разі наявності попередньої версії файлу, виконується порівняння.

В випадку знаходження попередньої версії виконуваного файла користувачу автоматично додатково відкривається вікно більш детальної інформацію про дослідження, яке представлено на рисунку 3.4.

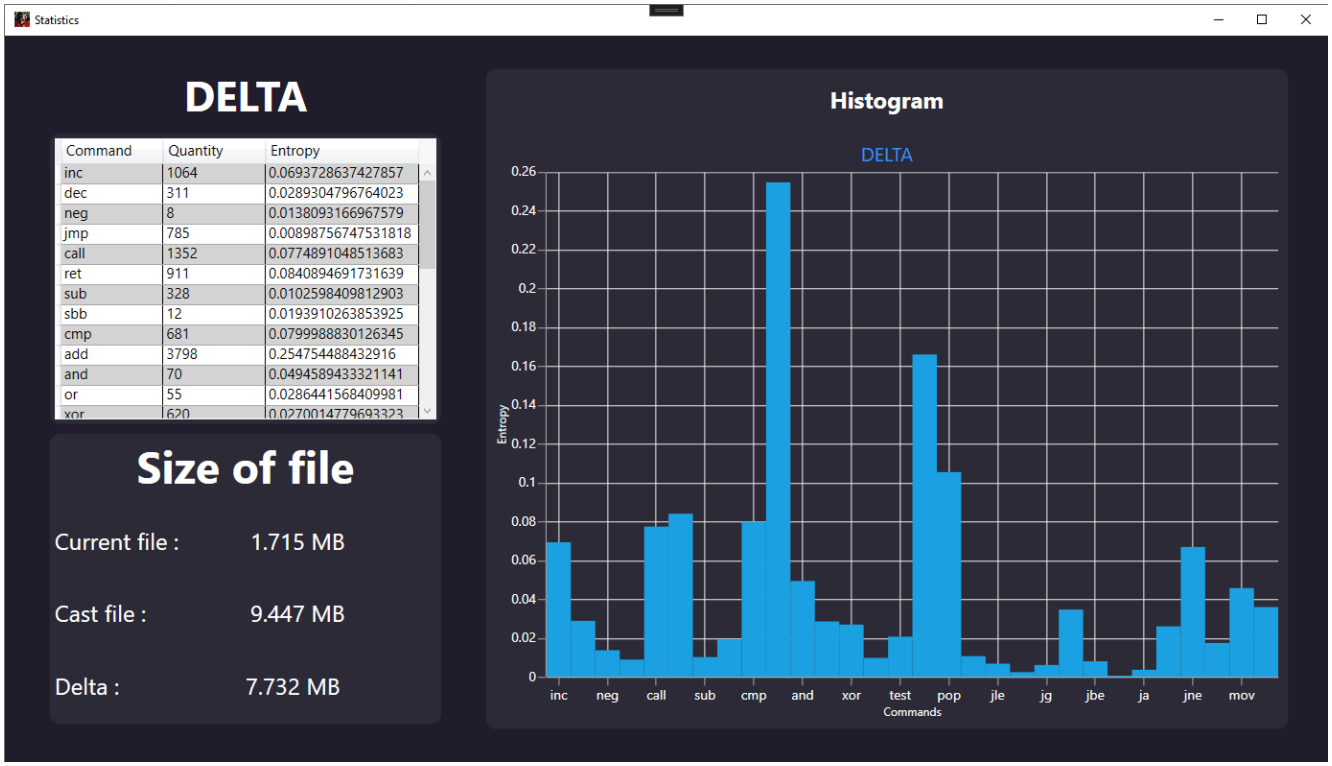

Рисунок 3.4 – При знаходженні попередньої версії виконуваного файла, користувач автоматично отримує додаткове вікно з детальною інформацією про дослідження.

Розмір програмного забезпечення

Програмне забезпечення «DisEn» складається з 1278 рядків коду, написаних мовами C# та XAML [21]. Розмір коду на диску становить приблизно 54.5 КБ Після компіляції та зборки, розмір «DisEn» становить 6 МБ.

Результати випробування програмного забезпечення

Програмне забезпечення «DisEn» було піддане випробуванням згідно програми та методикою випробувань наведених у додатку Д. Нижче наведені результати випробувань:

У результаті випробування функціональності програмного забезпечення «DisEn» підтверджено, що всі основні функції такі як дизасемблювання виконуваного файлу, перегляд вихідного тексту, розрахунок ентропії, представлення результатів у вигляді таблиці, представлення результатів у вигляді графіка, порівняння з попередньою версією, працюють належним чином.

4 ОХОРОНА ПРАЦІ

Охорона праці є комплексною системою заходів та засобів, спрямованих на збереження життя, здоров'я та працездатності працівників під час їх трудової діяльності. Ця система має ряд завдань, які вирішуються через інженерно-технічні та соціальні заходи.

Інженерно-технічні заходи охорони праці спрямовані на запобігання та мінімізацію ризиків для працівників. Це включає впровадження нових технологій, застосування безпечних матеріалів, розробку ергономічних робочих місць, встановлення систем вентиляції та освітлення. Такі заходи сприяють створенню безпечного та комфортного робочого середовища.

Соціальні заходи охорони праці спрямовані на відшкодування шкоди, завданої працівникам під час трудового процесу [22]. Це включає медичне обслуговування, реабілітацію, компенсаційні виплати, страхування, соціальну підтримку та захист прав працівників. Ці заходи сприяють відновленню та підтримці здоров'я працівників.

Охорона праці регулюється законодавством, зокрема в Україні діє Закон "Про охорону праці". Цей закон встановлює права та обов'язки працівників і роботодавців щодо охорони праці. Він визначає вимоги до організації робочого процесу, проведення оцінки ризиків, забезпечення безпеки та здоров'я працівників, надання медичної допомоги, розслідування нещасних випадків та багато іншого.

Нагляд та контроль є важливим елементом охорони праці. Державні органи, відповідальні за охорону праці, здійснюють перевірки підприємств, консультують та рекомендують заходи щодо безпеки та здоров'я працівників. Це сприяє дотриманню вимог охорони праці та запобігає порушенням.

Отже, охорона праці є системою заходів, спрямованих на забезпечення безпеки, здоров'я та працездатності працівників під час їхньої трудової діяльності. Вона базується на комплексі правових, соціальних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів та засобів.

Дотримання принципів охорони праці є важливим завданням кожного робочого місця для забезпечення безпеки та добробуту працівників.

4.1 Аналіз шкідливих та небезпечних факторів під час роботи з екранними пристроями

Аналіз шкідливих та небезпечних факторів під час роботи з екранними пристроями є важливою частиною охорони праці, оскільки тривалий час, проведений перед екраном, може викликати різні проблеми зі здоров'ям. Основні шкідливі та небезпечні фактори, пов'язані з роботою з екранними пристроями, включають:

– Вплив візуального навантаження: Постійна фокусування на екрані може призвести до втоми та напруги очей, сухості та подразнення очей, а також до розвитку офтальмологічних захворювань, таких як комп'ютерний синдром.

– Погіршення позиції тіла: Некоректна позиція тіла під час роботи з екраном може призвести до розладів опорно-рухового апарату, наприклад, до болей у шиї, плечах, спині, а також до м'язових напруг і захворювань суглобів.

– Вплив електромагнітного випромінювання: Екранні пристрої випромінюють електромагнітне випромінювання, яке може мати вплив на здоров'я працівників, включаючи можливість розвитку електромагнітної гіперчутливості.

– Погіршення психоемоційного стану: Тривала робота з екраном може спричиняти стрес, збільшену втомлюваність, розлади сну та інші психоемоційні проблеми.

– Регулювання робочого часу: Важливо встановити оптимальну тривалість роботи перед екраном і враховувати перерви для відпочинку та очей. Рекомендується регулярно проводити короткі перерви під час тривалої роботи з екраном.

– Забезпечення правильної позиції тіла: Рекомендується належно налаштувати робоче місце, включаючи висоту стільця та стола, кут нахилу екрана

і клавіатури. Привертайте увагу до правильної позиції спини, шиї і рук під час роботи перед екраном.

– Організація робочого простору: Важливо забезпечити належне освітлення робочого місця, уникати відблисків на екрані та регулювати контрастність та яскравість екрану. Крім того, слід стежити за правильним розміщенням робочого обладнання та аксесуарів.

– Застосування засобів індивідуального захисту: Врахуйте можливість використання антиблискових екранів, захисних окулярів або інших засобів індивідуального захисту для зменшення впливу шкідливих факторів.

– Свідоме використання часу поза робочим місцем: Забезпечте собі достатньо часу для фізичної активності, відпочинку і віддалення від екранів поза робочим часом. Це допоможе знизити загальний вплив тривалої роботи з екраном на організм.

– Проведення фізичних вправ: Після тривалої роботи за комп'ютером важливо робити паузи і проводити фізичні вправи для розслаблення м'язів і покращення кровообігу.

– Регулярні огляди зі спеціалістом: Проходження регулярних медичних оглядів та консультацій з офтальмологом може допомогти виявити можливі проблеми зі здоров'ям, пов'язані з роботою за екранними пристроями.

– Коректура робочого часу: Важливо розподіляти робочий час з екранними пристроями раціонально, з урахуванням перерв і відпочинку, щоб уникнути перенавантаження та втоми.

Норми мікроклімату зазначені у санітарних нормах мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [23].

Для аналізу шкідливих та небезпечних факторів під час роботи з екранними пристроями, слід провести оцінку ризику на робочих місцях, що включає:

– Визначення тривалості роботи з екраном та частоти перерв.

– Забезпечення оптимального освітлення: Важливо мати належне освітлення на робочому місці, щоб уникнути зайвого напруження очей. Краще

використовувати природне освітлення або штучне освітлення з регульованою яскравістю.

– Перевірка належності робочого місця до ергономічних стандартів, включаючи правильну позицію тіла, регулювання стільця та стола, використання ергономічних пристроїв.

– Оцінка впливу електромагнітного випромінювання та вжиття заходів для зниження можливих ризиків.

– Забезпечення навчання працівників з правил безпеки та користування екранними пристроями.

– Забезпечення належної вентиляції [24] приміщення: Наявність достатньої свіжого повітря у робочій зоні є важливим для підтримки комфортних умов та запобігання накопиченню шкідливих речовин.

– Встановлення акустичного комфорту: Звуковий комфорт є важливим аспектом безпеки та благополуччя працівників. Необхідно використовувати засоби для зменшення шуму, такі як акустичні екрани або навушники з шумозаглушенням.

– Правильне розташування обладнання: Клавіатура, миша та інші пристрої повинні бути розташовані так, щоб забезпечувати природні та комфортні рухи рук і запобігати напруженням м'язів і суглобів.

– Належне організування кабелів: Кабелі повинні бути правильно організовані, щоб уникнути спотикання, заплутування та інших потенційних небезпек.

Загалом, важливо підтримувати баланс та свідомо ставитись до роботи з екранними пристроями, дотримуючись принципів ергономіки та забезпечуючи безпечні умови праці для збереження здоров'я та працездатності. Врахування цих вимог допоможе зменшити вплив небезпечних та шкідливих факторів на здоров'я працівників, покращити комфорт роботи та забезпечити безпечні умови праці.

4.2 Методи нейтралізації або зменшення впливу негативних факторів під час роботи зі мобільними телефонами

Нейтралізація шкідливих та небезпечних факторів при роботі з мобільними пристроями може бути досягнута за допомогою наступних заходів:

1) Зменшення ефекту на зір:

– Регулярно робіть перерви під час роботи з мобільним пристроєм. Загалом рекомендується робити перерву кожні 20-30 хвилин і протягом 5-10 хвилин відводити погляд від екрану.

– Виконуйте вправи для очей, такі як погляд у далечину, обертання очей у різні напрямки і покладання теплих компресів на очі.

– Правильно налаштуйте яскравість, контрастність та розмір шрифту на своєму пристрої для забезпечення комфортного читання.

2) Менеджмент стресу:

– Встановіть часові обмеження для використання мобільних пристроїв і дотримуйтесь їх.

– Вимкніть сповіщення, коли вони необхідні, або встановіть режим "Не турбувати" під час важливих зустрічей або відпочинку.

– Займіться релаксаційними практиками, такими як медитація, глибоке дихання або йога, для зняття стресу і напруження.

– Збереження правильної пози:

– Підтримуйте правильну позу тіла під час використання мобільного пристрою. Завжди стежте за положенням голови, шиї та спини.

– Використовуйте підставку або тримайте пристрій на такій висоті, щоб ваша голова була в природному положенні і ви могли дивитися прямо на екран без похилення голови.

3) Зменшення впливу електромагнітного випромінювання:

– Використовуйте гарнітуру або вбудовані функції гучномовця, щоб зменшити близький контакт пристрою з головою під час розмов.

– Обмежте тривалість розмов по мобільному телефону і використовуйте функцію високоякісного звуку (HD Voice) для зниження рівня експозиції.

4) Керування залежністю від мобільних пристроїв:

– Встановіть правила для себе щодо використання мобільних пристроїв, наприклад, обмеження часу, проведеного з пристроєм, або встановлення безпосередніх "безмобільних" періодів.

– Підтримуйте здорові звички, такі як виконання фізичних вправ, соціальні контакти та заняття хобі, які не пов'язані з мобільними пристроями.

– Загалом, баланс і свідоме використання мобільних пристроїв, разом з регулярними перервами і здоровими звичками, допоможуть нейтралізувати більшість шкідливих та небезпечних факторів, пов'язаних з роботою з мобільними пристроями.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи була вирішена практична задача інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов – було успішно розроблено програмне забезпечення для знаходження внесених змін в код виконуваних файлів «DisEn». В ході виконання роботи були вирішені наступні завдання:

– Проведено аналіз практичної задачі інженерії програмного забезпечення включаючи аналіз предметної області та існуючих аналогів.

– Було сформульовано мету та постановку задачі на розробку.

– Були розроблено проект програмного забезпечення включаючи побудовані моделі варіантів використання та їх специфікації, архітектура.

– Було побудовано ескізний проект, технічний проект, робочий проект програмного забезпечення.

– Було проведено тестування та випробування програмного забезпечення.

У процесі розробки були використані такі інструменти: WPF, C#, DumpBin. Ці інструменти допомогли забезпечити функціональність програмного забезпечення. Розроблене програмне забезпечення "DisEn" допомагає знаходити зміни коду виконуваних файлів та робити висновки на предмет авторства цих змін за рахунок аналізу дизасембльованого коду виконуваних файлів, обчислення ентропії та порівняння з попередніми значеннями. Розроблена програма може бути використана, зокрема, для валідації існуючих файлів. В подальшому передбачається удосконалення розробленої програми за рахунок додавання математичної моделі для прогнозування авторства внесених змін. Також планується реалізувати можливість налаштування списку команд для розрахунку ентропії для більш гнучкого використання програми.

У рамках виконання кваліфікаційної роботи розроблено розділ з охорони праці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Каплун В.А., Майданюк В.П. Захист операційних систем. Навчальний посібник. Вінниця: ВНТУ, 2006. 180 с.

2. Що таке ін'єкція коду в Windows? URL: https://www.thefastcode.com/uk-uah/article/what-is-code-injection-on-windows (дата звернення 10.11.2022).

3. Бурячок В.Л., Аносов А.О., Семко В.В., Соколов В.Ю., Складанний П.М. Технології забезпечення безпеки мережевої інфраструктури. К.: КУБГ, 2019. 218 с.

4. Макарова Л.М., Камінський С.С., Бризгалов М.В. Використання ентропії для знаходження внесення змін коду виконуваних файлів. *Інформаційні технології: моделі, алгоритми, системи (ITMAS-2022): Матеріали III Всеукраїнської науково-практичної інтернет-конференції* (26-28 жовтня 2022 р.). Миколаїв: НУК імені адмірала Макарова, 2022. С.77-79.

5. Макарова Л.М., Камінський С.С., Бризгалов М.В. DISEN - a program for detecting changes made to the executable file code. *«FREE AND OPEN SOURCE SOFTWARE» (FOSS-2023): XIV-a Міжнародна науково-практична конференція «Free and Open Source Software», Харків,* (14-16 лютого 2023 р.). – Харків: Харківський національний економічний університет імені Семена Кузнеця, 2023. – С.15-16.

6. Макарова Л.М., Камінський С.С., Бризгалов М.В. Розробка програмного забезпечення для знаходження внесених змін в код виконуваних файлів. Вчені записки Таврійського національного університету імені В. І. Вернадського. Серія: Технічні науки. Київ: ТНУ, 2023. Том 34 (73) № 2. Частина 1. С.179-185. DOI https://doi.org/10.32782/2663-5941/2023.2.1/29

7. Энтропия. Как хаос помогает искать вирусы. URL: https://xakep.ru/2021/01/29/viruses-entropy/ (дата звернення 05.11.2022).

8. Shannon C.E., Weaver W. The Mathematical Theory of Communication. The University of Illinois Press, 1971. 144 p.

9. IDA Pro. URL: https://hex-rays.com/ida-pro/ (дата звернення 17.11.2022).

10. Immunity Debugger. URL: https://www.immunityinc.com/products/debugger/ (дата звернення 19.11.2022).

11. Capstone. The Ultimate Disassembler. URL: https://www.capstone-engine.org/ (дата звернення 21.11.2022).

12. Visual Studio URL: https://visualstudio.microsoft.com/ru/ (дата звернення: 25.05.2023)

13. DUMPBIM REFERENCE URL: https://learn.microsoft.com/en-us/cpp/build/reference/dumpbin-reference?view=msvc-170 (дата звернення 21.05.2023).

14. Методи інженерії програмного забезпечення URL: https://op.edu.ua/ru/education/programs/components/10441 (дата звернення: 20.05.2023)

15. Опорний конспект лекцій з курсу «Аналіз вимог до програмного забезпечення» для студентів напрямку підготовки «Програмна інженерія» URL: http://dspace.wunu.edu.ua/bitstream/316497/7281/1/FCIT_kKN_sPZS_dAVPZ_%20LEC.pdf (дата звернення: 20.05.2023)

16. Розділяй та володарюй: що таке патерни MVC і MVP, та як їх використовувати URL: https://highload.today/uk/blogs/shho-take-mvc-ta-mvp-patterni/ (дата звернення: 23.05.2023)

17. Простий посібник зі схем UML і моделювання баз даних URL: https://www.microsoft.com/uk-ua/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling (дата звернення: 19.05.2023)

18. МЕТОДОЛОГІЯ IDEF1X URL: https://stud.com.ua/87187/ekonomika/metodologiya_idef1x (дата звернення: 25.05.2023)

19. Руководство по WPF URL: https://metanit.com/sharp/wpf/ (дата звернення: 26.05.2023)

20. Язык C# и платформа .NET URL: https://metanit.com/sharp/tutorial/1.1.php (дата звернення: 26.05.2023)

21. Введение в язык XAML URL: https://metanit.com/sharp/wpf/2.php (дата звернення: 27.05.2023)

22. Про охорону праці. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 23.05.2023)

23. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#n14 (дата звернення: 23.05.2023)

24. Санітарні норми мікроклімату виробничих приміщень (ДСН 3.3.6.042-99): https://zakon.rada.gov.ua/rada/show/va042282-99#Text (дата звернення: 23.05.2023)

Додаток А – Технічне завдання на розробку програмного забезпечення для знаходження внесених змін в код виконуваних файлів

Вступ

Повна назва програмного продукту: програмне забезпечення для знаходження внесених змін в код виконуваних файлів «DisEn»

Сфера застосування:

Програмне забезпечення DisEn має широкий спектр застосувань у сферах, пов'язаних з аналізом виконуваних файлів та виявленням змін в коді. Деякі з основних сфер застосування включають:

— Інформаційна безпека: DisEn може бути використаний для виявлення змін в коді виконуваних файлів, що може свідчити про вторгнення або зловживання в програмне забезпечення. Це дозволяє підвищити рівень безпеки комп'ютерних систем і вчасно реагувати на потенційні загрози.

— Валідація програмного забезпечення: DisEn може використовуватись для перевірки цілісності виконуваних файлів та виявлення змін, які можуть бути результатом несанкціонованої модифікації або помилки під час розробки. Це дозволяє забезпечити якість та надійність програмного забезпечення.

— Авторське право та патентування: DisEn може служити інструментом для виявлення змін в коді виконуваних файлів та дослідження авторства. Це може бути корисно при розгляді судових справ, пов'язаних з порушенням авторських прав або патентних прав.

1. Підстави для розробки

Підставою для розробки даного програмного забезпечення є наказ на затвердження тем кваліфікаційних робіт бакалаврів зі спеціальності 121 «Інженерія програмного забезпечення» в Національному університеті кораблебудування ім. адмірала Макарова.

2. Призначення розробки

2.1 Функціональне призначення

Функціональне призначення програмного забезпечення "DisEn" полягає в автоматизації процесу виявлення та аналізу змін, які були зроблені у програмному коді під час розробки або після нього. Воно надає користувачам можливість порівнювати різні версії виконуваних файлів та виявляти зміни, що були внесені між ними.

Користувач програми може:

— Провести дизасемблювання та отримати вихідний код.

— Переглянути вихідний код в текстовому редакторі.

— Виконати розрахунок ентропії.

— Переглянути результати дослідження у табличній формі або у формі графіка.

— У разі знаходження попередньої верії файла з такою ж назвою виконати порівняння поточної та попередньої версії.

2.2 Експлуатаційне призначення

Експлуатаційне призначення програмного забезпечення «DisEn» полягає у знаходженні внесених змін в код виконуваних файлів.

3. Вимоги до програми або програмного виробу

3.1 Вимоги до складу виконуваних функцій

Функції користувача:

— Дизасемблювання та отримання вихідного коду: Здійснення процесу дизасемблювання, що дозволяє отримати вихідний код програми з виконуваного файлу. Це дозволяє розглянути внутрішню реалізацію програми.

— Перегляд вихідного коду в текстовому редакторі: Можливість відкрити отриманий вихідний код в текстовому редакторі для зручного перегляду, редагування та аналізу коду.

— Виконання розрахунку ентропії: Можливість виконати розрахунок ентропії для вихідного коду програми. Це допомагає оцінити складність та рівень випадковості коду.

— Перегляд результатів дослідження у табличній формі або у формі графіка: Подання результатів дослідження змін у виконуваному файлі у вигляді табличних даних або графіка. Це дозволяє візуалізувати та аналізувати зміни в зручному форматі.

— Порівняння поточної та попередньої версії файла з такою ж назвою: Перевірка наявності попередньої версії файла з такою ж назвою і можливість порівняти поточну версію з попередньою. Це дозволяє виявити зміни між версіями файлів і оцінити їх вплив.

3.2 Вимоги до організації вхідних та вихідних даних

Дані для аналізу в «DisEn» можуть бути подані у вигляді виконуваних файлів, які підлягають дизасемблюванню та аналізу.

Данні аналізу в «DisEn» можуть бути подані у табличному форматі та у вигляді графіка з можливостю скачування вихідного коду виконуваного файлу.

3.3 Вимоги до надійності

3.3.1 Вимоги до забезпечення надійного (сталого) функціонування програми.

Надійне (стійке) функціонування програми має бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік яких подано нижче:

— захист від некоректних дій користувача(під час розпізнавання);

— встановлення .NET Framework не нижче версії 4.5.

3.3.2 Час відновлення після відмови

Вимоги до часу відновлення після відмови не пред’являються.

3.3.3 Відмови через некоректні дії оператора

Відмова можлива при наступних умовах:

— Непередбачені вхідні дані: Програмне забезпечення «DisEn» повинно забезпечувати обробку непередбачених вхідних даних або файлів невідповідного формату. Якщо такі дані надходять на вхід, програма повинна надати користувачеві відповідне повідомлення про помилку та запитати коректні дані для продовження роботи.

3.4 Умови експлуатації

3.4.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні відповідати вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

3.4.2 Вимоги до чисельності та кваліфікації користувачів

Вимоги до чисельності користувачів для програмного забезпечення «DisEn» для знаходження внесених змін в код виконуваних файлів є гнучкими, і програма може успішно працювати з одним користувачем. Програмне забезпечення «DisEn» має певні вимоги щодо кваліфікації користувачів. Основні вимоги в цьому контексті включають:

— Базові знання програмування: Користувачі «DisEn» повинні мати базові знання програмування, включаючи розуміння основних понять, структур даних та алгоритмів. Це допоможе їм ефективно користуватись програмою та зрозуміти результати, які вона надає.

— Розуміння структури виконуваних файлів: Користувачам «DisEn» слід мати загальне розуміння структури виконуваних файлів, таких як виконуваний код, бібліотеки, символи тощо. Це допоможе їм краще розуміти виведені результати та здійснювати аналіз змін.

— Вміння працювати з інтерфейсом користувача: «DisEn» надає користувачам графічний інтерфейс, за допомогою якого вони можуть взаємодіяти з програмою. Користувачам слід мати базові навички роботи з графічними інтерфейсами, такі як навігація, введення даних та інтерактивні операції.

— Уміння аналізувати результати: Користувачі «DisEn» повинні мати уміння аналізувати результати, які надає програма, і здійснювати відповідні дії на основі цих результатів. Це може включати інтерпретацію знайдених змін в коді, співставлення їх з відповідними змінами вихідних файлів та прийняття відповідних рішень.

У загальному, користувачі «DisEn» повинні мати базові знання програмування, розуміння структури виконуваних файлів і вміння працювати з

графічним інтерфейсом. Крім того, важливо мати аналітичні навички для ефективного аналізу результатів, отриманих від програми.

3.5 Вимоги до складу і параметрів технічних засобів

Для роботи із програмою необхідний наступний склад технічних засобів з мінімальними параметрами:

— Процесор Intel Core i5 або еквівалентного рівня

— Оперативна пам'ять 4 ГБ.

— Дискового простору 100 МБ для встановлення програми та збереження результатів аналізу.

— Роздільна здатність екрану не менше 1280x800 пікселів.

3.6 Вимоги до інформаційної та програмної сумісності

Для коректної роботи програми «DisEn» на комп'ютері користувача необхідно мати наступне програмне забезпечення:

— Операційна система Windows: «DisEn» розроблена для роботи на операційній системі Windows. Тому користувач повинен мати встановлену сумісну версію Windows, наприклад, Windows 10, Windows 8 або Windows 7.

— .NET Framework: "DisEn" може вимагати наявності .NET Framework на комп'ютері користувача. .NET Framework – це технологічна платформа, яка дозволяє виконувати програми, написані на різних мовах програмування, включаючи C#, VB.NET та інші. Користувач повинен встановити відповідну версію .NET Framework, яка підтримується програмою «DisEn».

3.7 Вимоги до захисту інформації та програм

Вимоги до захисту інформації та програм не пред'являються

3.8 Спеціальні вимоги

Програма повинна забезпечувати взаємодію з користувачем за допомогою графічного інтерфейсу користувача, розробленого відповідно до рекомендацій компанії-виробника операційної системи.

4. Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

— технічне завдання;

— програму та методику випробувань;

— керівництво користувача;

— опис програми;

— код програми.

5. Техніко-економічні показники

У даній кваліфікаційній роботі техніко-економічні показники не розраховуються

6. Етапи роботи

Етапи роботи представлені у таблиці А.1

Таблиця А.1 – Етапи роботи

Номер	Назва етапів роботи	Терміни виконання
1.	Аналіз вимог до ПЗ	24.03.2023
2.	Розробка архітектури ПЗ	30.03.2023
3.	Розробка проекту ПЗ	04.04.2023
4.	Реалізація та тестування ПЗ	20.04.2023
5.	Випробування ПЗ	26.05.2023

7. Порядок контролю та приймання

В рамках розробки програмного забезпечення «DisEn», контроль за аналізом та проектуванням кожної окремої частини програми здійснюється на кожному етапі розробки з урахуванням вимог, що були визначені у технічному завданні. Кожна стадія розробки має бути завершена у встановлені строки та погоджена зі замовником.

Прийом програмного виробу здійснюється згідно з програмою і методикою випробувань. Під час прийому проводяться тестування, а результати фіксуються в

протоколі проведення випробувань. Якщо під час прийому виявляються помилки, складається акт про знайдені помилки, який підписують представники замовника і розробника, і затверджується керівником організації – замовника та організації – розробника.

Після отримання акта про знайдені помилки, розробник має не більше ніж 2 тижні на виправлення цих помилок. Після виправлення розробник повідомляє замовника про повторну перевірку. Процес виправлення помилок і повторної перевірки повинен бути документований і здійснюватись у встановлені строки.

Додаток Б – Вихідні тексти програмних модулів

Disassembler.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Runtime.Serialization.Formatters.Binary;
using System.Text;

namespace DisEn
{
   // Command info struct
   [Serializable]
   public class DisassemblerCommandInfo
   {
      public string Name { get; set; }
      public Int32 Count { get; set; }
      public double Entropy { get; set; }
   }

   // Class for disassembling
   [Serializable]
   public class Disassembler
   {
      #region Variables

      // File name
      private String _fileName;
      // Parsed file code
      private String _parsedFileCode;
      // File path to disassembled file
      [NonSerialized]
      private String _filePath;
      // Total instruction counter
      private Int32 _totalInstructionCounter;
      // Total entropy value
      private double _totalEntropyValue;
      // Size of the file in bytes
      private double _fileSize;
      // List of instructions
      private HashSet<string> _instructionFilterHashSet;
      // Dictionary of instructions
      private Dictionary<string, Int32> _instructionsDict;
      // Commands info list
      private List<DisassemblerCommandInfo> _disassemblerCommandsInfo;

      // Temporary folder for disassembler temp files
      [NonSerialized]
      private const string TEMP_DISASSEMBLER_FOLDER = "DISTEMP";
```

```
#endregion

#region Constructor

// Constructor
public Disassembler()
{
   // Initialize containers
   _instructionFilterHashSet = new HashSet<string>();
   _instructionsDict = new Dictionary<string, Int32>();
   _disassemblerCommandsInfo = new List<DisassemblerCommandInfo>();
   // Reset number to zero
   _totalInstructionCounter = 0;
}

#endregion

#region Methods

public override bool Equals(object obj)
{
   if (obj is Disassembler objectType)
   {
      return this._fileName.CompareTo(objectType._fileName) == 0
         && this._parsedFileCode.CompareTo(objectType._parsedFileCode) == 0
         && this._fileSize == objectType._fileSize
         && this._totalInstructionCounter == objectType._totalInstructionCounter
         && this._totalEntropyValue == objectType._totalEntropyValue
         && CompareDisInfo(objectType.GetDisassemblerCommandsInfo());
   }
   return false;
}

public override int GetHashCode()
{
   return base.GetHashCode();
}

// Set instruction hash set
public void SetInstructionFilterHashSet(HashSet<string> InstructionFilterHashSet)
{
   _instructionFilterHashSet = InstructionFilterHashSet;
}

// Get file name
public String GetFileName()
{
   return _fileName;
}

// Get file path
public String GetFilePath()
```

```
    {
        return _filePath;
    }

    // Get instruction hash set
    public HashSet<string> GetInstructionFilterHashSet()
    {
        return _instructionFilterHashSet;
    }

    // Total quantity of instructions
    public Int32 GetTotalInstructionCounter()
    {
        return _totalInstructionCounter;
    }

    // Total file size
    public double GetFileSize()
    {
        return _fileSize;
    }

    // Returns file entropy value
    public double GetFileTotalEntropyValue()
    {
        return _totalEntropyValue;
    }

    public List<DisassemblerCommandInfo> GetDisassemblerCommandsInfo()
    {
        return _disassemblerCommandsInfo;
    }

    public static void Serialize(String path, Disassembler disassembler)
    {
        BinaryFormatter binFormat = new BinaryFormatter();
        using (Stream fStream = new FileStream(path,FileMode.Create, FileAccess.Write,
FileShare.None))
        {
            binFormat.Serialize(fStream, disassembler);
        }
    }
    public static Disassembler Deserialize(String path)
    {
        BinaryFormatter binFormat = new BinaryFormatter();
        using (Stream fStream = new FileStream(path, FileMode.Open, FileAccess.Read,
FileShare.None))
        {
            return (Disassembler)binFormat.Deserialize(fStream);
        }
    }
```

```
        // Disassembles file
        public void DisassembleFile(String filePath)
        {
            // Save file path and its name
            FileInfo fileInfo = new FileInfo(filePath);
            _fileName = fileInfo.Name.Split('.')[0];
            _filePath = filePath;
            _fileSize = fileInfo.Length;
            // Disassemble and parse txt file
            if (fileInfo.Extension.CompareTo(".exe") == 0)
            {
                // Temp disassembler directory
                String _tempDisassembledFileFolder = TEMP_DISASSEMBLER_FOLDER;
                Directory.CreateDirectory(_tempDisassembledFileFolder);
                String _tempDisassembledFilePath = _tempDisassembledFileFolder + "\\" + _fileName +
".txt";
                // Disassemble file and parse it
                DisassembleExeToTxtFile(filePath, _tempDisassembledFilePath);
                ParseDisassembledTxtFile(_tempDisassembledFilePath);
                // Remove disassembled file and folder
                if (Directory.Exists(_tempDisassembledFileFolder))
                {
                    Directory.Delete(_tempDisassembledFileFolder, true);
                }
            }
            // Parse .txt file
            else if (fileInfo.Extension.CompareTo(".txt") == 0)
            {
                ParseDisassembledTxtFile(filePath);
            }
            else
            {
                return;
            }
            // Form and calculate data
            FormCommandsInfo();
            CalculateFileTotalEntropyValue();
        }

        // Exports disassembled file to .txt file
        public void ExportDisassembledCodeToTxtFile(String fileExportPath)
        {
            File.WriteAllText(fileExportPath, _parsedFileCode);
        }

        // Parses disassembled txt file
        private void ParseDisassembledTxtFile(String filePath)
        {
            if (File.Exists(filePath))
            {
                // Read all strings from file
```

```
            ParseDisassemble(File.ReadAllLines(filePath));
        }
    }

    // Disassembles file to get map of instructions (instruction, number of encounters)
    private void DisassembleExeToTxtFile(String filePath, String fileSavePath)
    {
        // Create dictionary of instructions
        _instructionsDict = new Dictionary<string, Int32>();
        // Open dumpbin
        ProcessStartInfo dumpInfo = new ProcessStartInfo();
        dumpInfo.FileName = @"Dumpbin\dumpbin.exe";
        dumpInfo.Arguments = String.Format(@"/disasm /out:{0} {1}", fileSavePath, filePath);
        Process dumpBinProcess = Process.Start(dumpInfo);
        // Check, if process still running
        try
        {
            while (Process.GetProcessById(dumpBinProcess.Id) != null) { }
        }
        catch (Exception ex) { }
    }

    // Forms commands info
    private void FormCommandsInfo()
    {
        // Initialize list with command and entropy value pair
        _disassemblerCommandsInfo = new List<DisassemblerCommandInfo>();
        List<string> instructionFilter = new List<string>(_instructionFilterHashSet);
        for (int i = 0; i < instructionFilter.Count; ++i)
        {
            // Check, if this element exist in map
            if (_instructionsDict.ContainsKey(instructionFilter[i]))
            {
                DisassemblerCommandInfo dissamblerCommandInfo = new
DisassemblerCommandInfo();
                dissamblerCommandInfo.Name = instructionFilter[i];
                dissamblerCommandInfo.Count = _instructionsDict[instructionFilter[i]];
                dissamblerCommandInfo.Entropy = EntropyCalculation(dissamblerCommandInfo.Count,
_totalInstructionCounter);
                // Count entropy of this element
                _disassemblerCommandsInfo.Add(dissamblerCommandInfo);
            }
        }
    }

    // Calculates file total entropy value
    public void CalculateFileTotalEntropyValue()
    {
        List<string> instructionFilter = new List<string>(_instructionFilterHashSet);
        // Calculate entropy
        _totalEntropyValue = 0;
        for (int i = 0; i < instructionFilter.Count; ++i)
```

```
        {
            // Check, if this element exist in map
            if (_instructionsDict.ContainsKey(instructionFilter[i]))
            {
                // Count entropy of this element
                _totalEntropyValue += EntropyCalculation(_instructionsDict[instructionFilter[i]],
_totalInstructionCounter);
            }
        }
    }

    // Parse disassemble
    private void ParseDisassemble(string[] dissFileLines)
    {
        StringBuilder codeStringBuilder = new StringBuilder();
        _totalInstructionCounter = 0;
        _instructionsDict.Clear();
        // Go through all instructions
        foreach (string line in dissFileLines)
        {
            // Add line to command string builder
            codeStringBuilder.AppendLine(line);
            foreach (string word in line.Split(' '))
            {
                if (_instructionFilterHashSet.Contains(word))
                {
                    // Add new instruction
                    if (!_instructionsDict.ContainsKey(word))
                    {
                        _instructionsDict.Add(word, 1);
                    }
                    else
                    {
                        // Increment instruction count
                        _instructionsDict[word]++;
                    }
                    // Increment total instruction counter
                    _totalInstructionCounter++;
                }
            }
        }
        // Copy string builder to string variable
        _parsedFileCode = codeStringBuilder.ToString();
    }

    // Counts entropy
    private double EntropyCalculation(double elementsCount, double allElementsCount)
    {
        if (allElementsCount == 0) { return 0; }
        return -elementsCount / allElementsCount * Math.Log(elementsCount / allElementsCount, 2);
    }
```

```
        private bool CompareDisInfo(List<DisassemblerCommandInfo> disComInfo)
        {
            if (_disassemblerCommandsInfo.Count != disComInfo.Count) { return false; }
            for (int i = 0; i < disComInfo.Count; ++i)
            {
                for (int j = 0; j < _disassemblerCommandsInfo.Count; ++j)
                {
                    if (_disassemblerCommandsInfo[j].Name.Equals(disComInfo[i]))
                    {
                        if (_disassemblerCommandsInfo[j].Count != disComInfo[i].Count
                            || _disassemblerCommandsInfo[j].Entropy != disComInfo[i].Entropy)
                        {
                            return false;
                        }
                    }
                }
            }
            return true;
        }

        #endregion
    }
}
```

DisassemblerComparator.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DisEn
{
    public class DisassemblerComparator
    {
        #region Variables

        // First disassembler
        Disassembler _firstDisassembler = new Disassembler();
        // Seconds disassembler
        Disassembler _secondDisassembler = new Disassembler();

        // Total entropy data
        private double _totalEntropyDelta = 0;
        // Total instruction counter
        private UInt32 _totalInstructionCounterDelta = 0;
        // Size of the file in bytes
        private double _fileSizeDelta = 0;
        // Commands delta
        private List<DisassemblerCommandInfo> _disassemblerCommandInfoDelta = new
List<DisassemblerCommandInfo>();
```

```
        #endregion

        #region Methods

        public Disassembler GetFirstDisassembler()
        {
            return _firstDisassembler;
        }

        public Disassembler GetSecondDisassembler()
        {
            return _secondDisassembler;
        }

        public double GetTotalEntropyDelta()
        {
            return _totalEntropyDelta;
        }

        public UInt32 GetTotalInstructionCounterDelta()
        {
            return _totalInstructionCounterDelta;
        }

        public double GetFileSizeDelta()
        {
            return _fileSizeDelta;
        }

        public List<DisassemblerCommandInfo> GetDisassemblerCommandInfoDelta()
        {
            return _disassemblerCommandInfoDelta;
        }

        // Return result data
        public bool CompareData(Disassembler firstDisassembler, Disassembler secondDisassembler)
        {
            if (firstDisassembler.Equals(secondDisassembler))
            {
                return true;
            }
            // Save disassembler objects
            _firstDisassembler = firstDisassembler;
            _secondDisassembler = secondDisassembler;
            // Calculate delta
            _totalEntropyDelta = Math.Abs(firstDisassembler.GetFileTotalEntropyValue() -
secondDisassembler.GetFileTotalEntropyValue());
            _totalInstructionCounterDelta =
(UInt32)Math.Abs(firstDisassembler.GetTotalInstructionCounter() -
secondDisassembler.GetFileTotalEntropyValue());
            _fileSizeDelta = Math.Abs(firstDisassembler.GetFileSize() -
secondDisassembler.GetFileSize());
```

```
            // Clear info array
            _disassemblerCommandInfoDelta.Clear();
            // Compare data
            for (int firstDisIndex = 0; firstDisIndex <
firstDisassembler.GetDisassemblerCommandsInfo().Count; ++firstDisIndex)
            {
               for (int secondDisIndex = 0; secondDisIndex <
secondDisassembler.GetDisassemblerCommandsInfo().Count(); ++secondDisIndex)
               {
                  if (firstDisassembler.GetDisassemblerCommandsInfo()[firstDisIndex].Name.Equals
                     (secondDisassembler.GetDisassemblerCommandsInfo()[secondDisIndex].Name))
                  {
                     if (firstDisassembler.GetDisassemblerCommandsInfo()[firstDisIndex].Count !=
secondDisassembler.GetDisassemblerCommandsInfo()[secondDisIndex].Count
                        || firstDisassembler.GetDisassemblerCommandsInfo()[firstDisIndex].Entropy !=
secondDisassembler.GetDisassemblerCommandsInfo()[secondDisIndex].Entropy)
                     {
                        DisassemblerCommandInfo disassemblerCommandInfo = new
DisassemblerCommandInfo();
                        disassemblerCommandInfo.Name =
firstDisassembler.GetDisassemblerCommandsInfo()[firstDisIndex].Name;
                        disassemblerCommandInfo.Count =
(Int32)Math.Abs(firstDisassembler.GetDisassemblerCommandsInfo()[firstDisIndex].Count -
secondDisassembler.GetDisassemblerCommandsInfo()[secondDisIndex].Count);
                        disassemblerCommandInfo.Entropy =
Math.Abs(firstDisassembler.GetDisassemblerCommandsInfo()[firstDisIndex].Entropy -
secondDisassembler.GetDisassemblerCommandsInfo()[secondDisIndex].Entropy);
                        _disassemblerCommandInfoDelta.Add(disassemblerCommandInfo);
                     }
                  }
               }
            }
            return false;
         }

         #endregion
      }
   }
```

DisassemblerManager.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;

namespace DisEn
{
   // Disassembler manager
   // Controls file saves and their management
   public class DisassemblerManager
   {
      #region Variables
```

```
// Current disassembler
private Disassembler _currentDisassembler = new Disassembler();

// Saved disassembler
private Disassembler _savedDisassembler = new Disassembler();

// Path to the instruction file
private const string INSTRUCTION_FILTER_FILE_PATH = "instructions.txt";

// Instruction filter
HashSet<string> instructionFilter;

#endregion

#region Constructor

public DisassemblerManager()
{
   // Prepare hash set
   instructionFilter = new HashSet<string>();
   // Get instruction filter list
   string[] instructionReadArray = File.ReadAllLines(INSTRUCTION_FILTER_FILE_PATH);
   for (int i = 0; i < instructionReadArray.Length; ++i)
   {
      // Split string
      string[] splitString = instructionReadArray[i].Split(' ');
      for (int j = 0; j < splitString.Length; ++j)
      {

         instructionFilter.Add(splitString[j]);
      }
   }
}

#endregion

#region Methods

public Disassembler GetCurrentDisassembler()
{
   return _currentDisassembler;
}

public Disassembler GetSavedDisassembler()
{
   return _savedDisassembler;
}

public bool IsSavedDisassemblerExistInData()
{
   return File.Exists(GetSavedDisassemblerPath());
```

```
        }

        public string GetSavedDisassemblerPath()
        {
            return String.Format("Data\\{0}\\{0}.asdat", _savedDisassembler.GetFileName());
        }

        public void SaveCurrentDisassemblerFile()
        {
            String directoryPath = String.Format("Data\\{0}", _savedDisassembler.GetFileName());
            if (!Directory.Exists(directoryPath))
            {
                Directory.CreateDirectory(directoryPath);
            }
            string dataDirectoryPath = String.Format("{0}\\{1}.asdat", directoryPath,
_savedDisassembler.GetFileName());
            if (File.Exists(dataDirectoryPath))
            {
                File.Delete(dataDirectoryPath);
            }
            Disassembler.Serialize(dataDirectoryPath, _currentDisassembler);
        }

        public void LoadSavedDisassemblerFile()
        {
            _savedDisassembler = Disassembler.Deserialize(GetSavedDisassemblerPath());
        }

        public void Disassemble(string filePath)
        {
            // Set instruction filter hash set
            _currentDisassembler.SetInstructionFilterHashSet(instructionFilter);
            _savedDisassembler.SetInstructionFilterHashSet(instructionFilter);
            // Disassemble files
            _currentDisassembler.DisassembleFile(filePath);
            _savedDisassembler.DisassembleFile(filePath);
            // Temp directory path
            if (!Directory.Exists("Temp"))
            {
                Directory.CreateDirectory("Temp");
            }
            string tempDirectoryPath = String.Format("Temp\\{0}.txt",
_currentDisassembler.GetFileName());
            // Export current file to temp
            _currentDisassembler.ExportDisassembledCodeToTxtFile(tempDirectoryPath);
            // Check, if file exist in the data directory
            if (IsSavedDisassemblerExistInData())
            {
                LoadSavedDisassemblerFile();
            }
            else
            {
```

```
            SaveCurrentDisassemblerFile();
        }
        // Open txt file
        if (File.Exists(tempDirectoryPath))
        {
            // Open file
            ProcessStartInfo txtInfo = new ProcessStartInfo();
            txtInfo.FileName = tempDirectoryPath;
            Process txtProcess = Process.Start(txtInfo);
        }
    }

    #endregion
  }
}
```

Додаток В – Опис програмного забезпечення

Вступ

"DisEn" є інструментом для знаходження внесених змін в код виконуваних файлів. Воно розроблене з метою спростити процес виявлення внесених змін та аналізу виконуваних файлів.

1. Основні функції

Програмне забезпечення "DisEn" має наступні основні функції:

1.1.Дизасемблювання виконуваних файлів: "DisEn" дозволяє користувачам дизасемблювати виконувані файли, отримувати доступ до їх вихідного коду та аналізувати його.

1.2.Розрахунок ентропії: Програма надає можливість розрахувати ентропію виконуваних файлів, що є важливим показником безпеки та надійності програмного коду.

1.3.Візуалізація результатів: "DisEn" надає зручні засоби візуалізації результатів аналізу, такі як таблиці та графіка, що спрощують сприйняття інформації та допомагають виявляти закономірності та тенденції.

1.4.Порівняння версій: Завдяки "DisEn" користувачам надається можливість порівняти різні версії виконуваних файлів та отримати результати порівняння для виявлення змін.

2. Архітектура

Програмне забезпечення "DisEn" базується на архітектурному шаблоні MVC (Model-View-Controller), що дозволяє ефективно організувати роботу з даними, відображенням та управлінням програмою.

3. Особливості

Особливості програмного забезпечення "DisEn" включають:

3.1.Висока швидкість обробки: Програма працює ефективно та швидко, забезпечуючи швидкість обробки великого обсягу виконуваних файлів.

3.2.Масштабованість: "DisEn" може працювати з різними типами виконуваних файлів та масштабуватися під потреби користувача.

3.3.Інтуїтивний інтерфейс користувача: Графічний інтерфейс програми призначений для зручної та простої взаємодії з користувачем, що спрощує процес аналізу виконуваних файлів.

3.4.Надійність та безпека: "DisEn" забезпечує надійність та безпеку обробки виконуваних файлів, що є критичними аспектами для програмного забезпечення, що працює з важливою інформацією.

4. Технічні засоби та програмне забезпечення, необхідне для функціонування програми

Програмне забезпечення повинна задовольняти вказаним вимогам на комп’ютері наступної мінімальної комплектації:

Для роботи з програмним забезпеченням DisEn рекомендовані наступні мінімальні параметри технічних засобів:

— Процесор Intel Core i5 або еквівалентного рівня

— Оперативна пам'ять 4 ГБ.

— Дискового простору 100 МБ для встановлення програми та збереження результатів аналізу.

— Роздільна здатність екрану не менше 1280x800 пікселів.

Для функціювання програмного забезпечення необхідно використовувати ОС Windows дистрибутів Window 7 та Visual Studio або іншу IDE (інтегроване середовище розробки) для роботи з мовою C# та .NET Framework або .NET Core для виконання програм.

Додаток Г – Інструкція користувача

Вступ

Ласкаво просимо до програмного забезпечення "DisEn"! Ця інструкція надасть вам детальну інформацію про використання програми та допоможе вам ефективно виконувати аналіз змін в коді виконуваних файлів. Будь ласка, ознайомтеся з усіма розділами цієї інструкції перед початком використання програми.

1. Вимоги до системи

Перед початком використання програмного забезпечення "DisEn" переконайтеся, що ваша система відповідає наступним вимогам:

— Процесор: Intel Core i5 або еквівалентний

— Оперативна пам'ять: Мінімум 4 ГБ

— Вільне місце на жорсткому диску: Мінімум 100 МБ

— Операційна система: Windows 7 або новіше

— Інтернет-підключення (для завантаження оновлень програми, якщо доступні)

2. Встановлення програми

Щоб встановити програму "DisEn", слідуйте цим крокам:

— Завантажте виконавчий файл встановлення програми з офіційного веб-сайту.

— Запустіть виконавчий файл і дотримуйтесь інструкцій на екрані для встановлення програми.

— Після завершення встановлення програма "DisEn" буде готова до використання.

— Використання програми

3. Завантаження виконуваного файлу

Для початку аналізу змін в коді виконуваного файлу слід виконати наступні кроки:

— Відкрийте програму "DisEn". Стартове вікно програми зображено на рисунку Г.1.

Рисунок Г.1 – Стартове вікно програми «DisEn»

— У головному меню виберіть опцію «Open File» в розділі опіцй «File». Головне вікно з вибраними опціями зображено на рисунку Г.2.

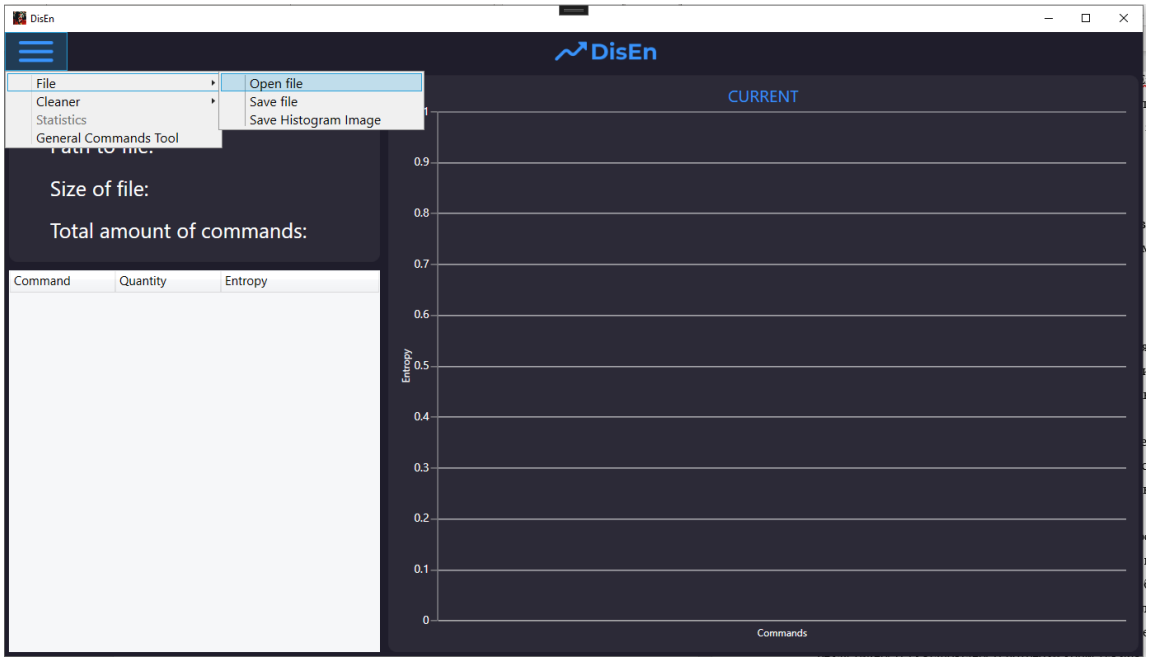

Рисунок Г.2 – Головне меню «DisEn» з вибраними опціями

— Виберіть потрібний виконуваний файл зі свого комп'ютера та натисніть "Відкрити". Діалогове вікно з вибором файлу наведено з на рисунку Г.3.

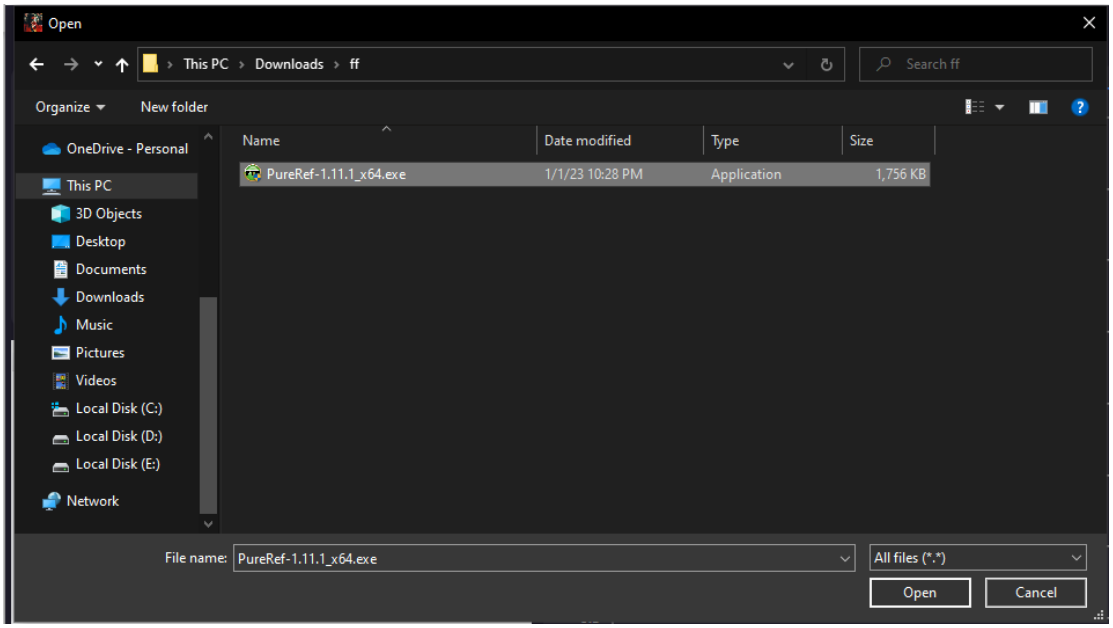

Рисунок Г.3 – Головне меню «DisEn»,вибір опції «Open File» в розділі опіцй «File»

4. Виконання аналізу

— Після вибору файла, програма автоматично почне аналізувати вибраний виконуваний файл.

— Зачекайте, поки програма завершить аналіз виконуваного файлу.

— Після результату аналізу програма відобразить результати на екрані, які наведено на рисунку Г.4.

Рисунок Г.4 – Завершення аналізу виконуваного файла

5. Перегляд результатів

Результати аналізу будуть відображені на екрані програми. Ви можете використовувати наступні функції для перегляду результатів:

— Вихідний текст виконуваного файла відкриється автоматично після завершення процесу аналізування. Вихідний текст виконуваного файлу зображено на рисунку Г.5.

Рисунок Г.5 – Вихідний текст виконуваного файла

— Для перегляду таблиці з даними аналізу потрібно звернути увагу на головне вікно програми «DisEn», таблиця буде заходитеся у лівому нижньому куті. Таблиця з даними аналізу зображена на рисунку Г.6.

Рисунок Г.6 – Відображення елемента інтерфейса «Таблиця» на головному вікні програмного забезпечення «DisEn»

— Для перегляду графіка з даними аналізу потрібно звернути увагу на головне вікно програми «DisEn», графік буде заходитеся у правій частині інтерфейсу. Графік з даними аналізу зображена на рисунку Г.7.

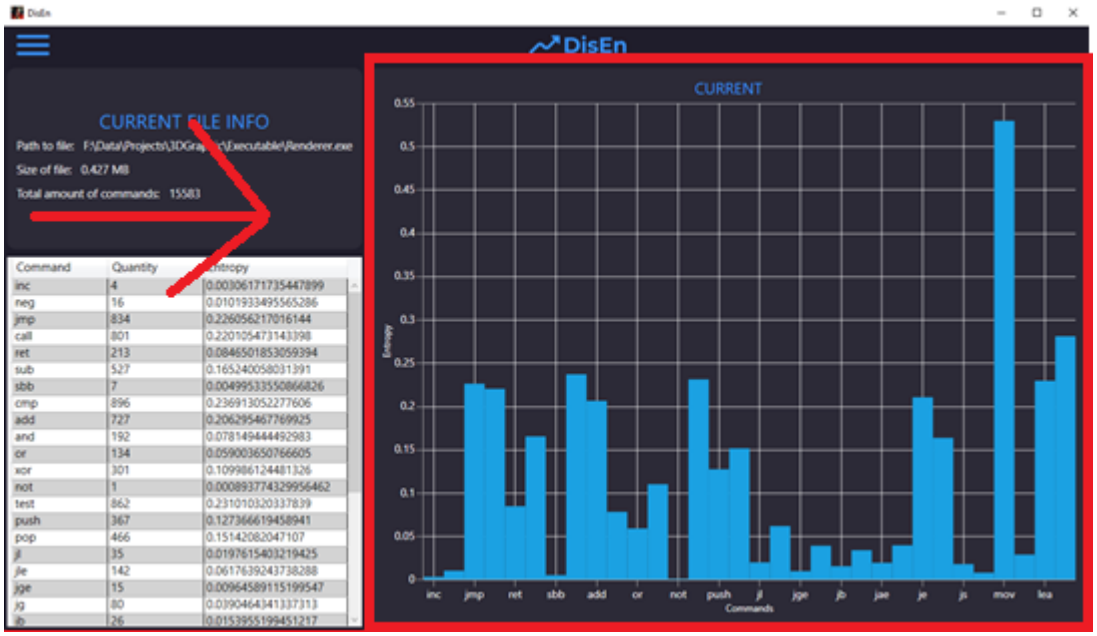

Рисунок Г.7 – Відображення елемента інтерфейса «Графік» на головному вікні програмного забезпечення «DisEn»

6. Порівняння версій (у разі наявності)

Якщо ви маєте попередні версії виконуваного файлу для порівняння, ви можете виконати наступні кроки:

— Виберіть потрібний виконуваний файл зі свого комп'ютера та натисніть "Відкрити".

— Після завантаження виконуваного файлу «DisEn» автоматично розпочне аналіз та порівняння двох версій файлу, якщо вона знайде зліпок аналізу попередньої версії файлу.

— Після завершення аналізу двох версій файлу, програма «DisEn» автоматично виведе результати на головне вікно програми. Крім того, вона відкриє вікно з більш детальною інформацією про аналіз виконуваних файлів. Вікна результату аналізу двох виконуючих файлів та вікно з більш детальною інформацією зображені на рисунках Г.8 та Г.9.

Рисунок Г.8 – Результат аналізу двох виконуваних файлів на головному вікні

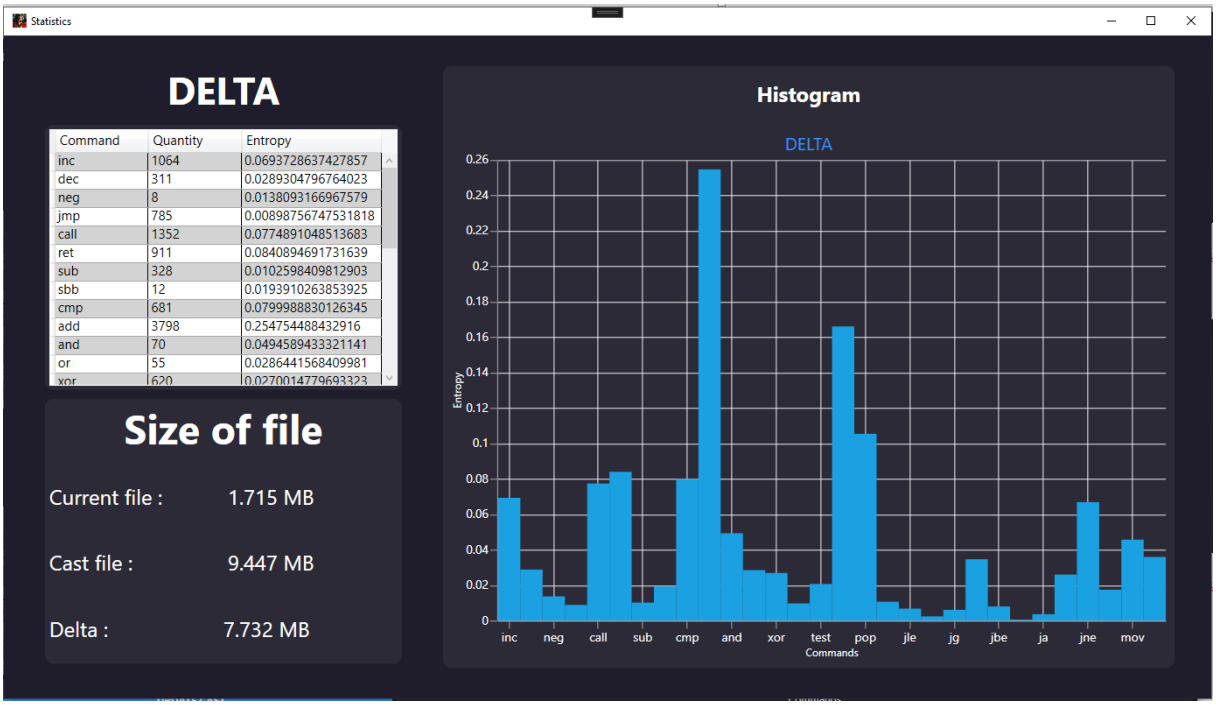

Рисунок Г.9 – Вікно з більш детальною інформацією про аналіз виконуваних файлів

Додаток Д – Програма та методика випробувань програмного забезпечення

1. Об’єкт випробування

Об'єктом випробування є програмне забезпечення "DisEn", яке розробляється для знаходження внесених змін у код виконуваних файлів. Ця розробка проводиться на основі потреб і вимог замовника з метою підвищення безпеки і захищеності інформаційних ресурсів.

2. Мета випробування

Метою випробування програми "DisEn" є перевірка її функціональності, надійності та відповідності вимогам замовника. Випробування спрямоване на виявлення можливих дефектів, помилок та проблем, які можуть вплинути на роботу програми та безпеку комп'ютерних систем.

Мета випробування полягає у забезпеченні якості програмного забезпечення, виявленні та виправленні дефектів та забезпеченні надійності та безпеки програми "DisEn".

Конкретні цілі випробування можуть включати:

1) Перевірка функціональності: Випробування включає перевірку правильності реалізації функцій програми "DisEn". Це означає переконання, що програма працює відповідно до очікувань, виконує необхідні завдання та надає очікувані результати.

2) Виконання реальних сценаріїв: Випробування може включати виконання реальних сценаріїв використання програми "DisEn" з метою перевірки її реалістичності та відповідності потребам користувачів.

3) Виявлення вразливостей: Випробування має на меті виявлення можливих вразливостей і слабких місць у програмних забезпеченнях, які можуть бути використані зловмисниками для несанкціонованого доступу або атак на інформаційні ресурси.

4) Тестування безпеки: Основна увага зосереджена на перевірці безпеки програми "DisEn" та виявленні можливих потенційних вразливостей. Це включає тестування на вразливості типу переповнення буфера.

3.Вимоги до програмного забезпечення

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

4. Склад програмної документації

Програмна документація повинна включати в себе наступні документи:

1) Технічне завдання

2) Текст програми.

3) Опис програми.

4) Інструкція користувача.

5) Програма та методика випробувань

5.Засоби і порядок випробувань

5.1 Технічні засоби, що використовуються під час випробувань

Склад технічних засобів, що використовуються під час випробувань:

– CPU: Intel Core i3 – 2.93 МГц;

– RAM: 4 GB;

- HDD: 2 GB вільного місця;

5.2 Програмні засоби, що використовуються під час випробувань

Склад програмних засобів, що використовуються під час випробувань:

- ОС Windows 10 x64 22H2 19045.2913;
- VisualStudio 22;
- пакет .NET Framework;

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

Представник служби, відповідальної за експлуатацію, виконує візуальну перевірку для забезпечення дотримання вимог програмної документації для програмного продукту. Під час перевірки порівнюється склад і комплектність програмної документації, наданої розробником, з переліком програмної документації, зазначеним у пункті "Склад програмної документації, пропонованої на випробування" цього документа.

Перевірка вважається завершеною, якщо склад та комплектність програмної документації, представленої розробником, відповідають переліку програмної документації, зазначеному у вищезазначеному пункті.

6.2 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Перевірка працездатності програми здійснюється відповідно до вимог до функціональних характеристик, зазначених у технічному завданні. Перевірка вважається завершеною, якщо склад та послідовність виконуваних дій відповідають вимогам до функціональних характеристик. Під час випробування була перевірена функціональність програми. В таблиці Д.1, наведеній нижче, вказано перелік тестів для основних функціональних можливостей.

Таблиця Д. 1- Результати випробування

№	Дія	Очікуваний результат	Результат перевірки
1	2	3	4
1	Дизасемблювання виконуваного файлу	Отримаємо файл формату .txt з вихідним кодом дизассембльованного виконуваного файлу	Програма успішно дизасемблювала виконуваний файл, вихідний код виконуваного файлу отримано
2	Перегляд вихідного тексту	Відображення коду у файлі формату .txt	Програма успішно відобразила текст вихідного коду

Продовження таблиці Д.1

1	2	3	4
3	Розрахунок ентропії	Числове значення ентропії для кожної команди з дизассембльованного виконуваного файлу	Програма успішно розрахувала ентропію для кожної команди з дизассембльованного виконуваного файлу
4	Представлення результатів у вигляді таблиці	Таблиця з даними, яка містить наступні поля: команди, кількість команд, ентропія команд	Програма успішно представила результати у вигляді таблиці. Строками таблиці є інформація про кожну команду
5	Представлення результатів у вигляді графіка	Графік, якій відображає данні про аналіз файлу	Програма успішно представила результати у вигляді графіка
6	Порівняння з попередньою версією	Виявлення різниць між двома версіями виконуваного файлу та відображення інформації про різницю у вигляді таблиці та графіку	Програма успішно порівняла виконуваний файл з попередньою версією та вивела інформацію про різницю у вигляді таблиці та графіку