ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ДЕННОЇ ОСВІТИ

ФОРМА НАВЧАННЯ ДЕННА

КАФЕДРА КОМП‘ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Допускається до захисту

Завідувач кафедри _____О.ОЛЬХОВСЬКА

(підпис)

«_____»_____2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОЇ РОБОТИ

на тему

Розробка програмного комплексу для виконання та оцінювання завдань з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів»

зі спеціальності 122 «Комп’ютерні науки»
освітня програма «Комп’ютерні науки»
ступеня магістра

Виконавець роботи: Фесенко Денис Ігорович

_____ «___»_____20__ р.

(підпис)

Науковий керівник к.ф.-м.н Олексійчук Юрій Федорович

_____ «___»_____20__ р.

(підпис)

ПОЛТАВА 2021 р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, символів, скорочень
Лог-файл	Місце для зберігається інформація про події у програмі.
Merge-Sort	Алгоритм сортування
Merge	Злиття
Парсинг	Синтаксичний аналіз
Хеш-функція	Функція, що перетворює вхідні дані будь-якого розміру в дані фіксованого розміру.
Sha-2	Secure Hash Algorithm Version 2 — безпечний алгоритм хешування, версія 2
GUI	Graphical user interface — графічний інтерфейс користувача

Вступ

Актуальність даної роботи полягає в створені програмного комплексу для виконання та оцінювання завдань для студентів та викладачів денної або дистанційної форми з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів».

Програмний комплекс має багато важливих пунктів актуальності використання, одні з основних є:

1) Два варіанта проходження прикладу: режим тренінгу та режим оцінювання.

2) Тренажер повідомляє про допущення помилки та надає підказку, якщо було зроблено багато помилок.

3) При проходження прикладу, є можливість переглянути попередні дії у лог-файлу

4) Після завершення трасування алгоритму в режимі оцінювання, з’являється вікно, з детальною інформацією про успіхи проходження прикладу і хеш-кодом та зберігаються у файл для подальшої перевірки викладачем.

5) Викладач має окрему програму для перевірки файлу студента на правдивість його результатів.

6) Програма повідомляє про будь-яку зміну файлу.

Мета даної роботи – створення програмного комплексу, який складається з двох програм. Перша програма допомагає студенту провести трасування алгоритму, щоб краще розібратися в роботі алгоритму та для вивчення основних його принципів та перевірити свої знання і отримати детальну інформації про успішність проходження. Друга програма допомагає викладачеві швидко перевірити файл студента на правдивість та дати остаточну оцінку.

Для досягнення мети були поставленні такі завдання:

- вивчення методичних рекомендацій та стандартів для оформлення та виконання магістерської роботи;

- ознайомлення з літературою про алгоритм сортування злиттям та хешування;
- інформаційний огляд;
- складання блок-схем для програми студента та викладача;
- розробка програмного комплексу;
- тестування двох програм.

Об’єктом розробки є програмний комплекс для дистанційного курсу «Аналіз алгоритмів».

Предметом розробки є розробка програмного комплексу для виконання та оцінювання завдання з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів».

Методи розробки. Для створення програмного комплексу було використано алгоритм сортування злиттям та хешування. Для розробки було застосовано об’єктно-орієнтовану мову програмування Java у вільному інтегрованому середовищі розробки Apache NetBeans.

Новизною даної роботи є програмний комплекс для виконання та оцінювання завдання з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів». При пошуку в інтернеті не було знайдено жодного програмного комплексу з такими функціоналом на дану тему.

Практичною цінністю програмного комплексу є дві програми для досконального вивчення або перевірки знань студента з теми «Аналіз алгоритму сортування злиттям» та швидка і якісна перевірка викладачем для остаточної оцінки знань студента.

Даний програмний комплекс рекомендовано використовувати студентами спеціальності «Комп’ютерні науки» дистанційного навчального курсу «Аналіз алгоритмів» в ПУЕТ. Він може бути впроваджений в дистанційний навчальний курс з «Аналіз алгоритмів» в Полтавському університеті економіки та торгівлі.

Пояснювальна записка складається з чотирьох розділів, а саме: постановка задачі, інформаційний огляд, теоретична та практична частина.

Пояснювальна записка містить: 62 сторінки, 39 рисунків, 1 таблиця, 1 додаток (на 66 сторінках) та 20 літературних джерел.

1 ПОСТАНОВКА ЗАДАЧІ

1.1 Постановка задачі розробки тренажера

Задачею даної роботи є програмний комплекс для виконання та оцінювання завдань з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів».

Для цього потрібно провести інформаційний огляд. Знайти та проаналізувати схожі програми та тренажери з даної теми або подібні по завданню.

Розглянути основні вимоги до роботи.

Основними вимогами для розробки програмного комплексу є:

1. Скласти алгоритми для двох програм.
2. Скласти блок-схеми відповідно до обраних алгоритмів.
3. Програмно реалізувати дві програми.
4. Перевірити працездатність усього програмного комплексу.

Основними вимогами до роботи та використання програмного комплексу:

1. Інтерфейс програми повинен бути зрозумілим для студенту та викладача.
2. Перед проходженням прикладу, реалізувати введення персональних даних.
3. При кожному введені або вибору відповіді, реалізувати перевірку даних та надати студентові інформацію про помилку і в разі багаточисленних помилок на одне запитання отримати підказку.
4. Відстежувати усі попередні кроки у лог-файлу під час проходження прикладу.
5. Після завершення проходження прикладу реалізувати підсумок проходження прикладу та зберігання його у файл для подальшої перевірки викладачем.

6. Реалізувати швидку та якісну перевірку файлу студента викладачем, та в разі зміни результатів повідомити викладача.

2 ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Огляд робіт, де розглянуте аналогічне до теми завдання

Під час проведення аналізу схожих тренажерів або програмних комплексів , були розглянуті статті, бакалаврські та магістерські роботи студентів, які схожі до поставленої задачі даної роботи, а саме Белінська В.В. [1], Жайворонок Я.І [3], Глущенко К.С., Гавриленко О.В. [4], Пасєка М.С., Пасєка Н.М., Бестильний М.Я., Шекета В.І. [14]. Дані роботи та статті було порівняно з висунутими вимогами.

Програмне забезпечення тренажера Белінська В.В. [1] з теми «Розподіли дискретних випадкових величин та їх числові характеристики» дистанційного навчального курсу «Теорія ймовірностей та математична статистика». Для реалізації програмного забезпечення було обрано середовище Microsoft Visual Studio та мову програмування C#. При запуску тренажера з’являється початкова форма тренажеру має дві функції перевірки знань: теоретична і практична, також є корисні доповнення у вигляді теоретичної відомості та мови інтерфейсу. При виборі одного режиму перевірки знань з’являються питання на які потрібно відповісти власноруч або вибрати відповідь. В програмі присутня перевірки на різні помилки такі як: неправильна відповідь та якщо відповідь було не введено або не обрано і вразі багатьох помилок програма дасть підказку. Тренінг завершується повідомленням про проходження та можливістю почати спочатку або закрити програму.

Програмне забезпечення Жайворонок Я.І. [3] для тренажера дистанційного навчального курсу «Теорія інформації та кодування» з теми «Коди із виявленням помилок». Для створення тренажеру було обрано середовище Unity, об'єктно–орієнтовної мови програмування C# та редактору коду Microsoft Visual Studio. При запуску тренажера відображується вікно, де наведена інформація про автора, та три кнопки: “Почати”, “Мова” і “Вихід”. Перед тренінгом користувач може змінити мову за необхідністю. Після натиснення кнопки “Почати”, з’явиться меню, де користувач зможе вибрати одну тему із списку усіх тем і напроти кожної з тем відображається оцінка, якщо користувач вже пройшов цю тему. Після того, як було

обрано тему, користувач може вибрати режим проходження: теоретичний або практичний. Обравши режим, з’явиться вікно на якому буде розташовано такі елементи інтерфейсу як: поле з питанням, поля для введення відповіді або для вибору відповіді, стовбець зі списком питань та три кнопки “Допомога”, “Відповісти” і “Закінчити”. Під час проходження тренінгу присутня перевірка введених або обраних відповідей та якщо користувач не зміг відповісти правильно програма дасть підказку. Також гарним доповненням є можливість переглянути свою оцінку в реальному часі. Тренінг завершується повідомленням про проходження і оцінкою та можливістю спробувати тренування на іншому пристрої, вийти в меню програми або закрити програму.

Аналіз методів порівняння текстів Глущенко К.С., Гавриленко О.В.[4] у випускних роботах студентів. В даній статі розглядається різноманітні методи та їх використання у роботах студентів. Було детально розглянуто принципи порівняння текстів, класифікація та проведено їх аналіз. Умови застосування методів та як правильно обрати метод для поставленої задачі.

Аналіз використання високоефективної реалізації функцій хешування SHA-512 Пасєка М.С., Пасєка Н.М., Бестильний М.Я., Шекета В.І. [14] для розробки програмних систем. В даній статті було розглянуто детально про хеш-функцію SHA-512 а саме: структурну схему і блок-схему алгоритму хешування, проблеми використання, програмна реалізація та доцільність використання у різних програмних продуктів.

В рамках бакалаврської роботи [2] розробка тренажеру з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів». Дана магістреська робота є маштабним доповненням та перобкою бакалаврської роботи. Чим саме відрізняється дана робота від попередної:

1. Тренажер має новий режим “Режим оцінювання”. Після запуску цього режиму користувачу потрібно буде ввести персональні данні.
2. Окрім практичних запитань було додано теоритичні питання

3. В кінці тренінгу програма надасть детальну інформацію про успішність виконная задачі користувачем, такі як: кількість помилок, час виконання, тощо.
4. Вся детальна інформація буде зберігатись у окремому файлі і матиме власну хеш-функцію.
5. Також до самої програми для перевірки знань студента було розроблено програма для викладача, яка зможе перевірити файл з хеш-функцією на правдивість результатів.

2.2 Позитивні аспекти оглянутих робіт

Переваги програмного забезпечення Белінська В.В. [1]:

1. Програмне забезпечення реалізовано на трьох мовах.
2. Можливість звернутися до теоретичних відомостей.
3. Присутня перевірка знань теоретична та практична.
4. Перевірка кожного вибору або введення відповіді та надання підказки при помилках.

Переваги тренажера Жайворонок Я.І. [3]:

1. Використання тренажеру на різних платформах.
2. Програмне забезпечення реалізовано на трьох мовах
3. Присутність різних тем на вибір
4. Можливість звернутися до лекційного матеріалу.
5. Перевірка кожного вибору або введення відповіді та надання підказки при помилках.
6. Можливість переглянути попередні питання та наступні.
7. Можливість відстежувати оцінку в реальному часі та в кінці тренінгу.

2.3 Вади розробок з оглянутих робіт

Недоліки тренажеру Белінська В.В. [1]:

1. Відсутні перегляду попередні дії.
2. Відсутність інформації про успішність проходження тренінгу

Недоліки тренажеру Жайворонок Я.І. [3]:

1. Дизайн тренажеру потрібно оновити
2. Для отримання лекційного матеріалу потрібно мати доступ у вихід в інтернет.

2.4 Необхідність та актуальність теми роботи

Необхідність створення тренажеру з теми «Аналіз алгоритму сортування злиттям», полягає у тому, що студенти, які навчаються дистанційно або заочно не мають таких можливостей у навчанні та перевірці своїх знань, як студенти які навчаються на денній формі. Тренажер призначений для часткової заміни викладача при вивчені відповідної теми.

3 ТЕОРЕТИЧНА ЧАСТИНА

3.1 Алгоритмізація задачі за темою роботи

Програма для студента побудована на алгоритмі сортування злиття, яка заснована на техніці розділяй і володарюй. Алгоритм цього типу працює, поділяючи велике завдання на дрібніші, які потім просто виконати. При сортуванні злиттям масив поділяється навпіл поки кожна ділянка не стане завдовжки в один елемент. Потім ці ділянки повертаються на місце (об’єднуються) в правильному порядку.

Розглянемо роботу алгоритму на прикладі:

Взято несортований масив:

Рисунок 3.1 – Несортований масив

Ділимо масив на рівні половини до тих пір, поки він не розділиться на максимально короткі масиви рис. (3.2-3.4):

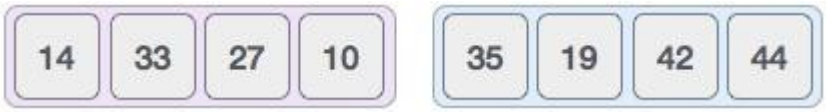

Рисунок 3.2 – Поділ на два масиви

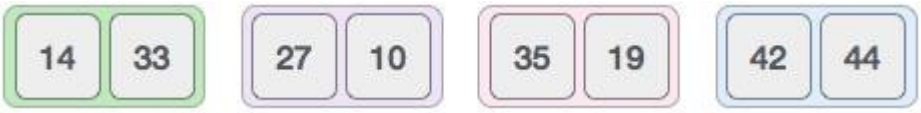

Рисунок 3.3 – Поділ на чотири масиви

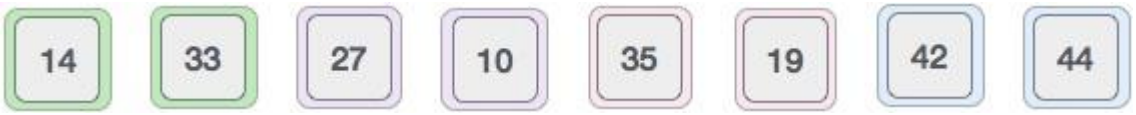

Рисунок 3.4 – Максимально короткі масиви

Тепер, коли початковий масив поділений на максимально короткі масиви, поєднуємо їх у правильному порядку. Спочатку порівнюємо елемент для кожного списку а потім об’єднуємо їх в інший список у відсортований спосіб. В результаті ми отримаємо спочатку масив по два відсортованих елементів, потів масив по чотири елементи і наприкінці повністю відсортований масив рис. (3.4-3.7)

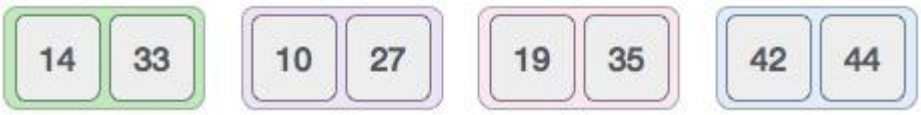

Рисунок 3.5 – Масив по два відсортованих елементів

Рисунок 3.6 – Масив по чотири відсортованих елементів

Рисунок 3.7 – Повністю відсортований масив

Для алгоритму роботи програми ми реалізуємо наступні операції:

1) Операцію для рекурсивного поділу масиву на групи (метод Sort).

```
Merge-Sort (A, p, r)
1       if p < r
2       q = [(p + r) / 2]
3       MERGE-SORT(A, p, q)
4       MERGE-SORT(A, q + 1, r)
5       MERGE(A, p, q, r)
```

2) Злиття в правильному порядку (метод Merge).

```
Merge (A, p, q, r)
1       n1 = q - p + 1
2       n2 = r - q
3       let L[1..n1 + 1] and R[1..n2 + 1) be new arrays
4       for i = ... to n1
5       L[i] = A[p + i - 1]
6       for j = ... to n2
7       R[j] = A[q + j]
8       L[n1 + 1] = ∞
9       R[n2 + 1] = ∞
10      i = 1
11      j = 1
12      for k = p to r
13      if L[i] <= R[j]
14      A[k] = L[i]
15      i = i + 1
16      else A[k] = R[j]
17      j = j + 1
```

Розглянемо приклад застосування алгоритму сортування злиттям і візьмемо один з прикладів тренажеру:

Приклад: A={12,4,8,5}

Далі йде поділ масиву на групи (метод Sort).

Merge-Sort(A,1,4)

1. If 1<4 true

2. Q=[(1+4/2)] =2

3. Merge-Sort(A,1,2)

3.1 if 1<2 true

3.2 q=[(1+2)/2] =1

3.3 Merge-Sort(A,1,1)

3.3.1 if 1<1 false

3.4 Merge-Sort(A,2,2)

3.4.1 if 2<2 false

Після поділу масиву на групи, здійснюється злиття перших двох чисел в правильному порядку (метод Merge).

3.5 Merge (A,1,1,2)

3.5.1 n1=1-1+1=1

3.5.2 n2=2-1=1

3.5.3 let L[1,2] and R[1,2] be new arrays

3.5.4 for i=1 to 1

3.5.5 L[1]= A[1]=12

3.5.4 for i=2 to 1 false

3.5.6 for j=1 to 1

3.5.7 R[1]=A[2]=4

3.5.6 for j=2 to 1 false

3.5.8 L[2] = ∞

3.5.9 R[2] = ∞

3.5.10 i=1

3.5.11 j=1

3.5.12 for k = 1 to 2

3.5.13 if L[1]<= R[1], 12<=4 false

3.5.16 else A[1]=R[1]=4

3.5.17 j=j+1=2

3.5.12 for k = 2 to 2

3.5.13 if L[1]<= R[2], 12<= ∞true

3.5.14 A[2] = L[1]=12

3.5.15 i=i+1=2

3.5.12 for k = 3 to 2 false

Знову ділимо масив на групи (метод Sort).

4.Merge-Sort(A,3,4)

4.1 if 3<4 true

4.2 q=[(3+4)/2]=3

4.3 Merge-Sort(A,3,3)

4.3.1 if 3<3 false

4.4 Merge-Sort(A,4,4)

4.4.1 if 4<4 false

Здійснюється злиття двох останніх чисел в правильному порядку (метод Merge).

4.5 Merge(A, 3,3,4)

4.5.1 n1= 3-3+1=1

4.5.2 n2=4-3=1

4.5.3 let L[1,2] and R[1,2] be new arrays

4.5.4 for i=1 to 1

4.5.5 L[1]= A[3]=8

4.5.4 for i=2 to 1 false

4.5.6 for j=1 to 1

4.5.7 R[1]= A[4]=5

4.5.6 for j=2 to 1 false

4.5.8 L[2] = ∞

4.5.9 R[2]= ∞

4.5.10 i=1

4.5.11 j=1

4.5.12 for k = 3 to 4

4.5.13 if L[1]<= R[1], 8<=5 false

4.5.16 else A[3]=R[1]=5

4.5.17 j=j+1=2

4.5.12 for k = 4 to 4

4.5.13 if L[1]<= R[2], 8<= ∞ true

4.5.14 A[4] = L[1]=8

4.5.15 i=i+1=2

4.5.12 for k = 5 to 4 false

Здійснюється злиття усіх чисел в правильному порядку (метод Merge).

5. Merge(A,1,2,4)

5.1 n1=2-1+1=2

5.2 n2=4-2=2

5.3 let L[1,3] and R[1,3] be new arrays

5.4 for i=1 to 2

5.5 L[1]= A[1]=4

5.4 for i=2 to 2 true

5.5 L[2]= A[2]=12

5.4 for i=3 to 2 false

5.6 for j=1 to 2

5.7 R[1]=A[3]=5

5.6 for j=2 to 2 true

5.7 R[2]=A[4]=8

5.6 for j=3 to 2 false

5.8 L[2] = ∞

5.9 R[2] = ∞

5.10 i=1

5.11 j=1

5.12 for k = 1 to 4

5.13 if L[1]<=R[1] 4<=5 true

5.14 A[1]=L[1]=4

5.15 i=1+1=2

5.12 for k=2 to 4

5.13 if L[2]<=R[1] 12<=5 false

5.16 else A[2]=R[1]=5

5.17 j=1+1=2

5.12 for k = 3 to 4

5.13 if L[2]<=R[2] 12<=8 false

5.16 else A[3]=R[2]=8

5.17 j=2+1=3

5.12 for k = 4 to 4

5.13 if L[2]<=R[3] ,12<=∞ true

5.14 A[4]=L[2]=12

5.15 i=2+1=3

5.12 for k = 5 to 4 false

Завершення сортування.

Під час проходження тренінгу, у програми присутній алгоритм для обрахування кількості помилок. Студент, який проходить приклад, може вписати або вибрати правильну відповідь, але якщо відповідь неправильна з'являється повідомлення про помилку і це є сигналом для програми, що було зроблено помилку. Під час усього тренінгу програма буде збирати помилки з кожного кроку або параметру, але якщо помилка була зроблена в одному і тому ж місці, наприклад напевному кроці було дано чотири поля для введення відповіді і якщо користувач впише в друге поле декілька разів не правильну відповідь, програма буде сприймати, що це одна помилка а не декілька. Також програма буде рахувати відсоток правильних відповідей при виборі кожного кроку або вписуванні і вибору параметру з кожного поля. Усе це можна буде побачити після закінчення тренінгу в кінцевій формі рис. (3.8)

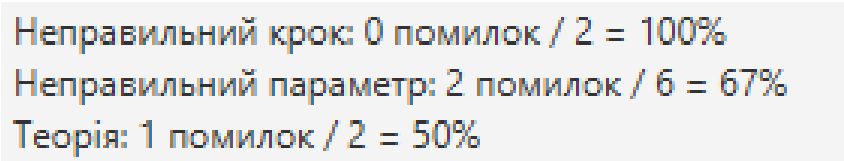

Рисунок 3.8 – Кількість помилок та відсоток правильних відповідей

Програма для студента після проходження створює файл с результатами проходженням та хеш-функцією. Хеш-функцією називають математичний алгоритм, що перетворює довільний масив даних у рядок фіксованої довжини, що складається з літер і цифр. Причому за умови використання того ж типу хешу довжина залишатиметься незмінною, незалежно від обсягу вступних даних.

Розглянемо роботу хеш-функції на прикладі:

Візьмемо довільне слово, наприклад: "Робота", після перетворення хеш-функцією SHA-1, буде виглядати так: 0b174d7312677ea86e0e409583eb40400a7f93fa. Тепер візьмемо теж саме слово, але з малої літери "робота" і знову використаєм перетворення хеш-функцією і вигляд буде таким: 98a9cb5aa3b704fca4f941916935fbbb7ce0bc12.

Як ми вже бачимо, результати відрізняються один від одного, навіть коло різниця між цими слова тільки у тому, з якої літери вони пишуться, з малої чи великої. Але між ними все ж таки є одна спільна особливість: кожен рядок має довжину рівно 40 символів. Тобто, будь-яке слово або текст буде хешуватися у рядок, який буде складатися рівно з 40 символів, а якщо детальніше, то 1128 символів, включаючи пробіли, будуть утиснуті до рядка тієї ж довжини, що й шестизначне слово.

Це потрібно для того, щоб контролювати цілісності важливих файлів операційної системи, важливих програм або даних. В нашому випадку, викладач зможе контролювати результати, щоб їх не змінювали студенти після проходження тренінгу.

Для реалізації хеш-функції, було обрано алгоритм SHA-2 – назва односторонніх хеш-функцій SHA-224, SHA-256, SHA-384, SHA-512 рис (3.9). Хеш-функція потрібна для формування “відбитків” або “дайджестів” повідомлень випадкової бітової довжини. Використовуються в різних додатках або компонентах, пов'язаних з захистом інформації.

Приклад:

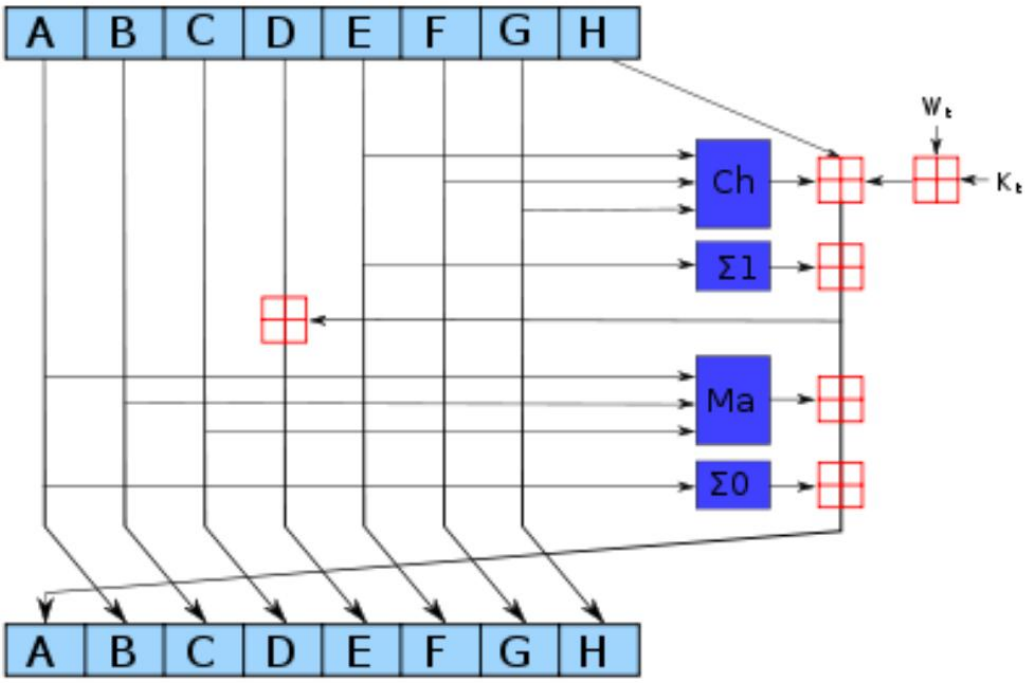

Рисунок 3.9 - Схема однієї ітерації алгоритмів SHA-2

Повідомлення після доповнення поділяється на блоки, кожен блок – на 8 слів. Алгоритм пропускає кожен з блоків повідомлень через цикл з 64 або 80 ітераціями. На кожній ітерації 2 слова з восьми перетворюються, функцію перетворення задають інші слова. Результати обробки кожного блоку складаються, сума є значенням хеш-функції. В таблиці показано більш детально деякі технічні характеристики різних варіантів SHA-2 рис.(3.1).

Хеш-функція	Довжина дайджесту повідомлення (біт)	Довжина внутрішнього стану (біт)	Довжина блоку (біт)	Максимальна довжина повідомлення (біт)	Довжина слова (біт)	Кількість ітерацій у циклі
SHA-256/224	256/224	256	512	$2^{64} - 1$	32	64
SHA-512/384	512/384	512	1024	$2^{128} - 1$	64	80

Таблиця 3.1 – Таблиця с технічними характеристиками

3.2 Розробка блок-схем алгоритмів

Дана блок-схема була побудована на алгоритмі роботи режиму оцінювання (блок-схему для режиму тренінгу можна переглянути у бакалаврській роботі [2]) для програми студента:

Крок 1. Запуск програми.

Крок 2. Вибір режиму.

Крок 3. Користувач вводить особисті данні

Крок 4. Користувач отримує випадковий приклад.

Крок 5. Програма має скінчену кількість запитань, якщо вони закінчилися (так), програма переходить на крок 10, якщо питання не закінчилися (ні), переходимо на крок 6.

Крок 6. Задається питання: який крок алгоритму?

Крок 7. Користувач вводить відповідь. Якщо відповідь правильна, переходимо на крок 7, якщо відповідь не правильна, з'являється повідомлення про помилку і поява підказки при багаторазових помилок. Користувач заново вводить відповідь.

Крок 8. Задається питання: яка буде відповідь даного кроку?

Крок 9. Користувач вводить відповідь. Якщо відповідь правильна, програма повертається до кроку 5, якщо відповідь не правильна, з'являється повідомлення про помилку і поява підказки при багаторазових помилок. Користувач заново вводить відповідь.

Крок 10. Користувач отримує тестові запитання.

Крок 11. Програма має скінчену кількість тестових запитань, якщо вони закінчилися (так), програма переходить на крок 14, якщо питання не закінчилися (ні), переходимо на крок 12.

Крок 12. Задається тестове питання.

Крок 13. Користувач вводить відповідь. Якщо відповідь правильна, переходимо на крок 10, якщо відповідь не правильна, з'являється повідомлення про помилку і поява підказки при багаторазових помилок. Користувач заново вводить відповідь.

Крок 14. Закінчення тренінгу і поява детальної інформації.

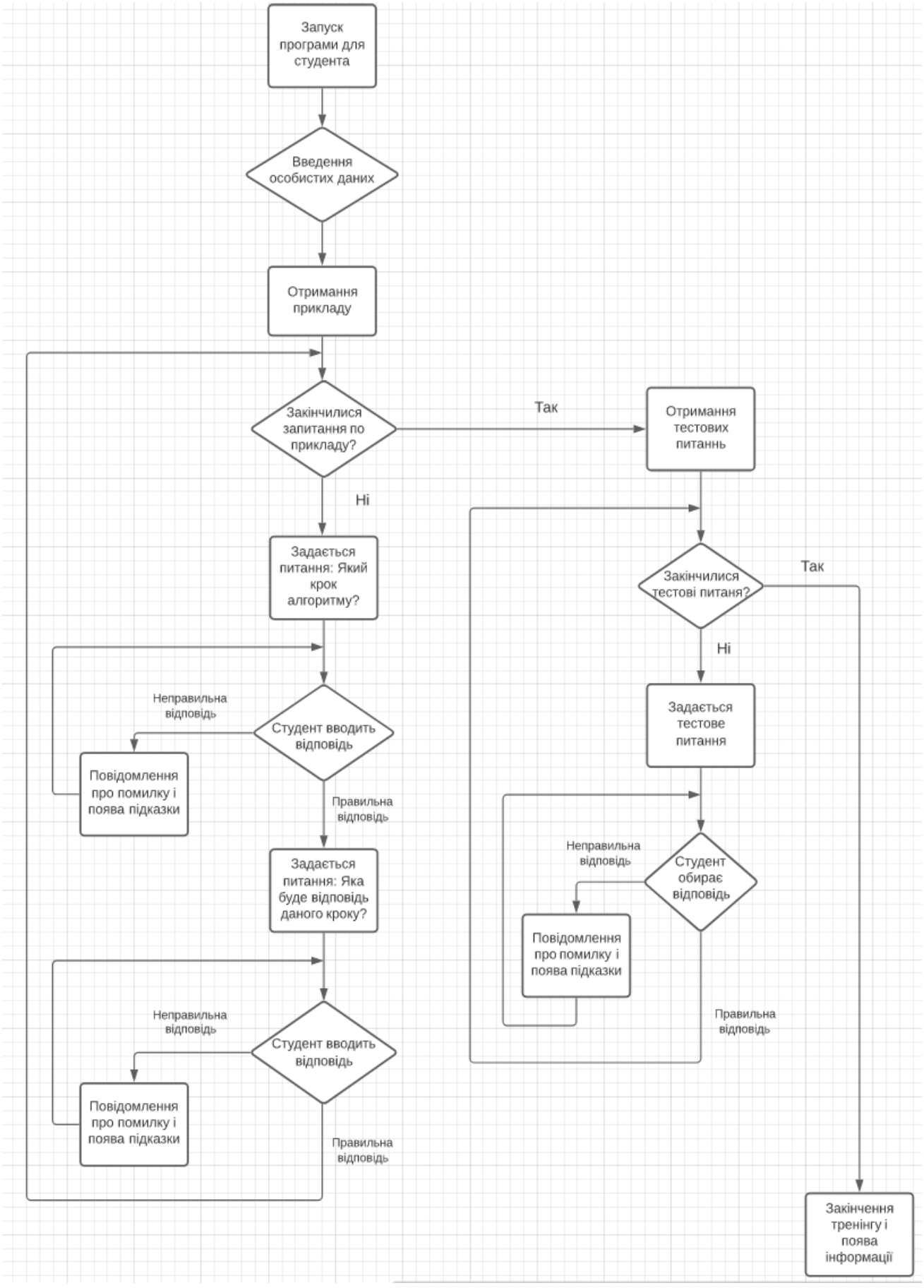

Рисунок 3.10 – Блок-схема алгоритму програми студента

Блок-схема для програми викладача:

Крок 1. Запуск програми.

Крок 2. Користувач вибирає файл з результатами.

Крок 3. Програма перевіряє файл на зміну результатів або файлу.

Крок 4. Результат або файл змінено? Якщо змінено (так), з’являється повідомлення про зміну результатів або файлу, якщо не змінено (ні), з’являється повідомлення про вірність результатів.

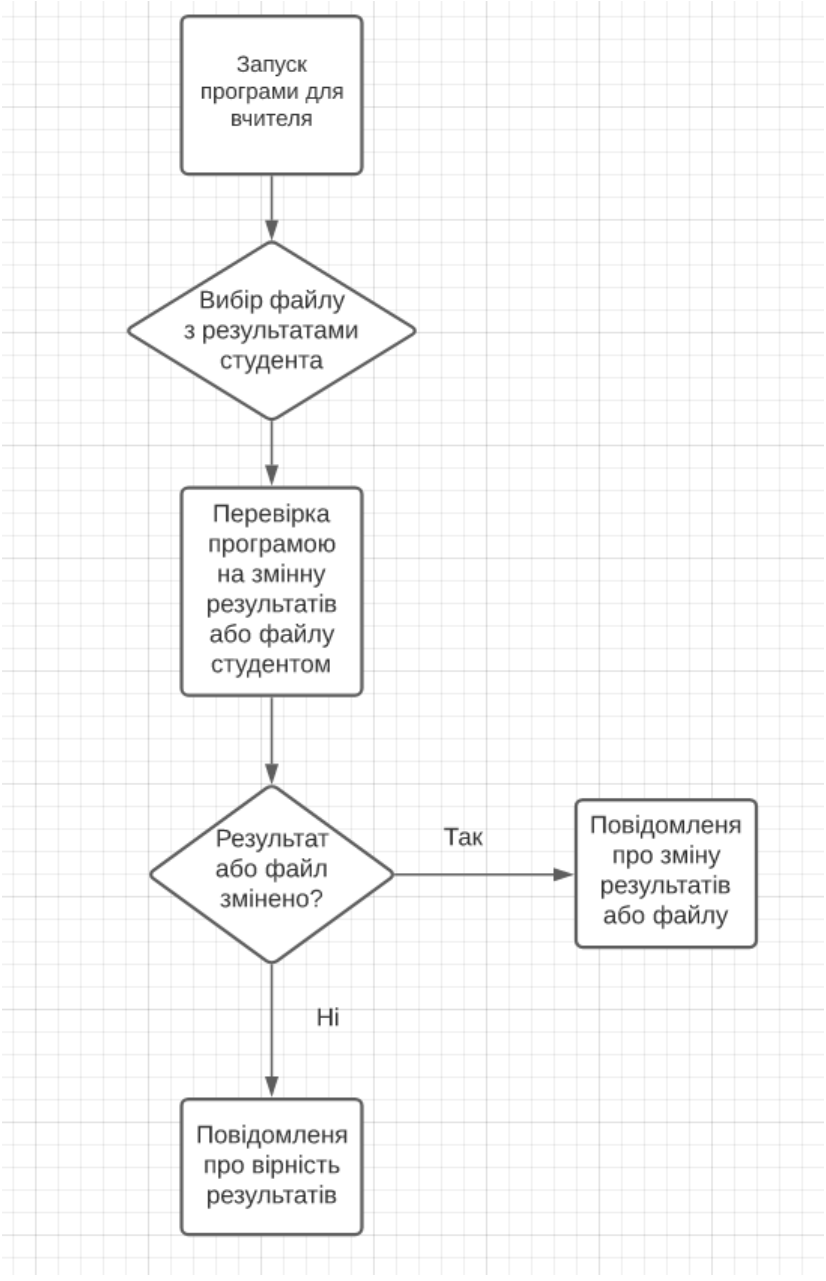

Рисунок 3.11 – Блок-схема алгоритму програми викладача

3.3 Обґрунтування вибору програмних засобів для реалізації завдання роботи.

Програмні засоби для реалізації програми студента.

JavaFX – набір інструментів для створення кросплатформенних графічних додатків на платформі Java.

Особливості JavaFX:

JavaFX насамперед поставляється з великим набором елементів графічного інтерфейсу, такі як: кнопки, таблиці, меню, діаграми, тощо. Що в свою чергу дає змогу сберегти досталь часу.

JavaFX часто використовує стилі CSS, а також є можливість використовувати спеціальні параметри FXML з метою формування GUI, щоб не роботи це в коді Java. Це полегшує швидке розташування графічного інтерфейсу користувача або зміна зовнішнооого вигляду.

JavaFX має готові до використання частини діаграми, тому не потрібно їх писати с самого нуля, коли потрібна базова діаграма.

JavaFX додатково поставляється з підтримкою 3D графіки, яка може бути корисна при розробці ігор або схожих додатків.

Програмні засоби для реалізації програми викладача.

Java MessageDigest – клас передбачає шифрувальну хеш-функцію, що може вичислювати дайджест повідомлення з бінарних даних. Коли ми отримуємо набір зашифрованих даних, то не маємо жодної гарантії, що він не був змінений. Дайджест повідомлення допомагає вирішити цю проблему.

Для того, щоб встановити чи було змінено дані, відправник повинен розрахувати дайджест повідомлення з даних і відправити їх разом с даними. Одержувач після отримання зашифрованих даних і дайджест повідомлення, може перерахувати дайджест повідомлення з даних і перевірити , чи відповідає обчислений дайджест повідомлення дайджесту повідомлення, отриманому з даними.

Якщо два дайджест повідомлення співпадають, то є велика ймовірність того, що результати були не змінені.

4 ПРАКТИЧНА ЧАСТИНА

4.1 Опис процесу програмної реалізації

Для виконання поставленої задачі було обрано об’єктно-орієнтовану мову програмування Java із застосуванням вільного інтегрованого середовища розробки Apache NetBeans.

Роботу розпочато зі створення файлу FXML, рис. (4.1-4.2).

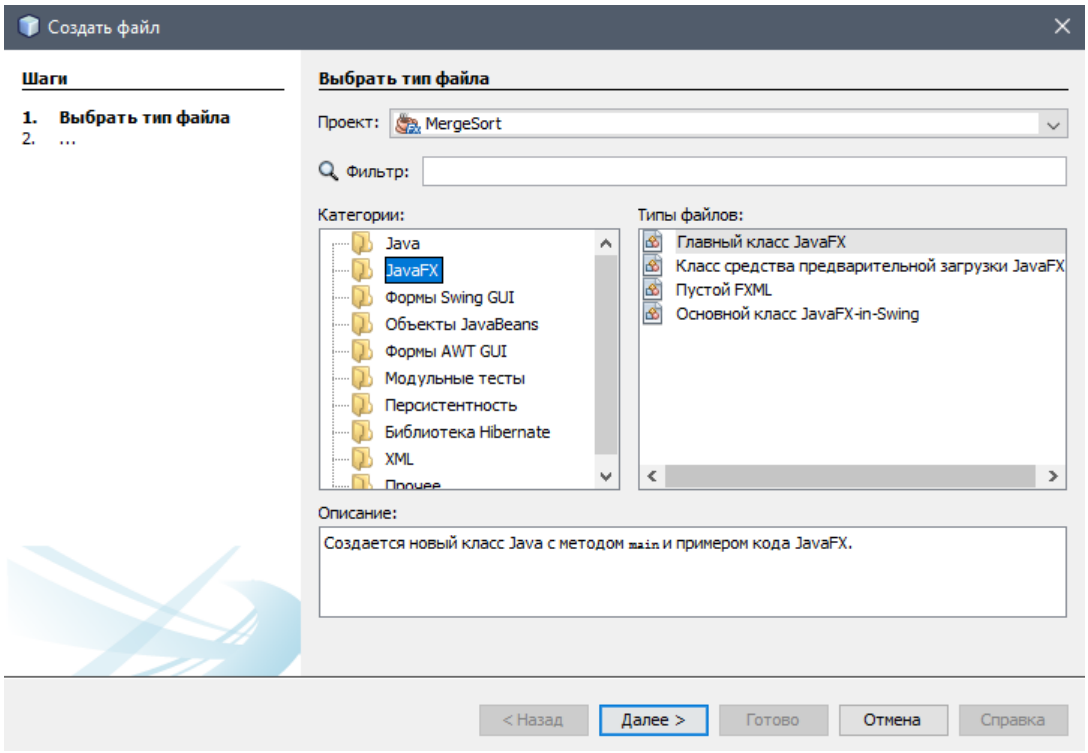

Рисунок 4.1 – Вибираємо тип файлу

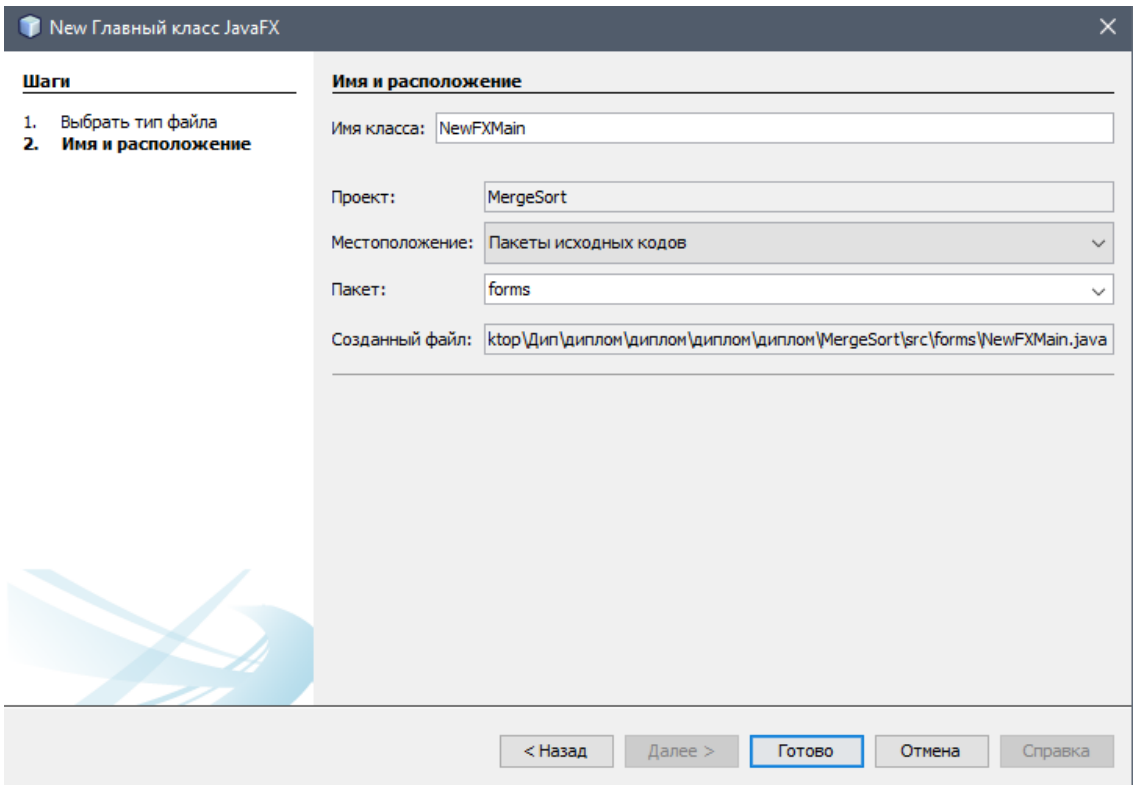

Рисунок 4.2 – Вибираємо ім’я та місце розташування.

Після створення файлу для програми студента було додано пакети, які потрібні для створення інтерфейсу. Для цього було підключено такі пакети:

<?import javafx.geometry.Insets?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.ComboBox?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.Separator?>

<?import javafx.scene.control.TextArea?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.BorderPane?>

<?import javafx.scene.layout.FlowPane?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.layout.VBox?>

<?import javafx.scene.text.Font?>

Під час розробки було створено 4 форми MainFormVersion1.fxml, MainFormVersion2.fxml, UserDataForm.fxml та WelcomeForm.fxml.

Приклад створення елементів у UserDataForm.fxml:

1. Створення стовбців та їх місце розташування на формі

<columnConstraints>

<ColumnConstraints hgrow="SOMETIMES" maxWidth="250.0" minWidth="2.0" prefWidth="97.0" />

<ColumnConstraints hgrow="SOMETIMES" maxWidth="436.0" minWidth="0.0" prefWidth="326.0" />

</columnConstraints>

<rowConstraints>

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

<RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />

</rowConstraints>

2. Створення полів для внесення особистих даних та їх місце розташування на формі

<children>

<TextField fx:id="nameField" GridPane.columnIndex="1" />

<TextField fx:id="surnameField" GridPane.columnIndex="1" GridPane.rowIndex="1" />

```
            <TextField fx:id="groupField" GridPane.columnIndex="1"
GridPane.rowIndex="2" />
            <TextField fx:id="keyField" GridPane.columnIndex="1"
GridPane.rowIndex="3" />
            <Label text="Ім'я" />
            <Label text="Прізвище" GridPane.rowIndex="1" />
            <Label text="Ключове слово" GridPane.rowIndex="2" />
            <Label text="Група" GridPane.rowIndex="3" />
 </children>
 <padding>
            <Insets bottom="10.0" left="20.0" right="20.0" top="20.0" />
 </padding>
```

3. Створення кнопки “Розпочати” та її місце розташування на формі

```
        Button fx:id="startButton" mnemonicParsing="false"
onMouseClicked="#goToMainFormVersion2" text="Розпочати" />
      <padding>
        <Insets bottom="20.0" />
      </padding>
```

Далі було створено 4 контролери WelcomeFormController.java, MainFormComtrollerVersion1.java, MainFormComtrollerVersion2.java, та UserDataFormController.java.

Приклад методів в WelcomeFormController.java та UserDataFormController.java:

1. Метод, який викликається при натисканні кнопки “Розпочати”

```
void goToMainFormVersion2(MouseEvent event) throws IOException {}
```

2. Метод, який викликається при натисканні кнопки “Перевірка результатів”

```
void goToHashCheckForm(MouseEvent event) throws IOException{}
```

3. Метод, який викликається при натисканні кнопки “Режим тренінгу”

void goToMainFormVersion1(MouseEvent event) throws IOException {}

4. Метод, який викликається при натисканні кнопки “Режим оцінювання ”

void goToUserDataForm(MouseEvent event) throws IOException {}

Також містить в собі текстовий формат JSON для передачі структурованої інформації через мережу.

Для підключення JSON було встановленно бібліотеки:

import org.json.simple.JSONArray;

import org.json.simple.JSONObject;

import org.json.simple.parser.JSONParser;

import org.json.simple.parser.ParseException;

Та створенний метод, який зчитує структуровану інформацію с JSON файлу.

private void readFile() {}

Також було створено файл Utils, який потрібний для хешування.

Для створення хешфункції було створено Utils та підключено такі бібліотеки:

import java.nio.charset.StandardCharsets;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

Розглянемо детально два методи для реалізації логіки хешування:

1. Метод для перетворення необроблених байтів хешу в рядкове Хеш-подання.

```
public static String toHexString(byte[] bytes) {
        StringBuilder hexString = new StringBuilder();

        for (int i = 0; i < bytes.length; i++) {
            String hex = Integer.toHexString(0xFF & bytes[i]);
            if (hex.length() == 1) {
                hexString.append('0');
            }
```

```
            hexString.append(hex);
        }

        return hexString.toString();
    }
```

2. Метод для отримання хешу SHA-512 з отриманих даних.

```
public static byte[] getHash(String data) throws NoSuchAlgorithmException {
        MessageDigest md = MessageDigest.getInstance("SHA-512");
        md.update(SALT);
        byte[] hash = md.digest(data.getBytes(StandardCharsets.UTF_8));

        return hash;
    }
```

Розглянемо детально декілька методів для логіки роботи програми для оцінювання:

1. Метод, який рахує кількість помилок, час початку, власні дані, тощо.

```
private String writeResultInFile() throws IOException, NoSuchAlgorithmException {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy
HH:mm:ss");
        int allParamsCount = 0;
        for (int i = 0; i < steps.length; i++) {
            allParamsCount += steps[i].getAnswers().length;
        }

        double rightStepsPercent = (wrongStepsCount * 100.0 / steps.length);
        double rightParamsPercent = (wrongParamsCount * 100.0 / allParamsCount);
        double rightTestsPercent = (wrongTestsCount * 100.0 / tests.length);
```

```
        StringBuilder builder = new StringBuilder();
        builder.append("Ім'я: " + studentName + "\n");
        builder.append("Прізвище: " + studentSurname + "\n");
        builder.append("Група: " + studentGroup + "\n");
        builder.append("Ключове слово: " + studentKey + "\n");
        builder.append("Процент пройденого: 100%" + "\n");
        builder.append("Неправильний крок: " + wrongStepsCount + " помилок / "
+ steps.length + " = "
                + Math.round(100 - rightStepsPercent) + "%\n");
        builder.append("Неправильний параметр: " + wrongParamsCount + "
помилок / " + allParamsCount + " = "
                + Math.round(100 - rightParamsPercent) + "%\n");
        builder.append("Теорія: " + wrongTestsCount + " помилок / " + tests.length
+ " = "
                + Math.round(100 - rightTestsPercent) + "%\n");
        builder.append("Час початку: " + formatter.format(startedAt) + "\n");
        builder.append("Час кінця: " + formatter.format(endedAt));

        String hash = Utils.toHexString(Utils.getHash(builder.toString()));
        builder.append("\n" + "Геш: " + hash);

        File file = new File(
                studentName + "_" +
endedAt.toInstant(ZoneOffset.ofTotalSeconds(0)).toEpochMilli() + ".txt");
        file.createNewFile();

        BufferedWriter writer = new BufferedWriter(new FileWriter(file));
        writer.write(builder.toString());
```

```
        writer.close();

        return builder.toString();
}
```

2. Метод для отримання індекса вибраного шагу алгоритму.

```
        private int getSelectedStepIndex() {
                if (selectedPart.equals("MERGE-SORT")) {
                        return mergeSortParts.indexOf(selectedStep);
                }

                return mergeParts.indexOf(selectedStep);
        }
```

3. Метод для перевірки чи пусті поля.

```
private boolean checkFields() {
        boolean isFieldsNotEmpty = true;
        for (Control c : fields) {
                if (c instanceof TextField) {
                        TextField f = (TextField) c;
                        isFieldsNotEmpty = isFieldsNotEmpty && f.getText() !=
null && !f.getText().isEmpty();
                }
        }
        return selectedPart != null && selectedStep != null &&
isFieldsNotEmpty;
    }
```

4. Метод для перірки відповідей

```
if (!checkFields()) {
                return false;
        }
```

```
            Step current = steps[stepIndex];
            if (!(current.getPart().equals(selectedPart)) || current.getStep() !=
        getSelectedStepIndex()) {
        счетчик неправильно
                if (countOfTries == 0) {
                    wrongStepsCount++;
                    wrongParamsCount += current.getAnswers().length;
                }

                return false;
            }
```

Після створення файлу для програми викладача було додано пакети, які потрібні для створення інтерфейсу. Для цього було підключено такі пакети:

```
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
```

Під час розробки було створена форма HashCheckForm.fxml

Приклад створення елементів у HashCheckForm.fxml:

1. Створення поля з прокруткою та її місце розташування.

```
<ScrollPane fitToHeight="true" fitToWidth="true" prefHeight="342.0"
prefWidth="787.0">
```

```
        <content>
          <TextArea fx:id="textArea" editable="false" prefHeight="325.0" prefWidth="785.0">
            <font>
              <Font size="18.0" />
            </font>
          </TextArea>
        </content>
        <VBox.margin>
          <Insets left="10.0" right="10.0" />
        </VBox.margin>
      </ScrollPane>
```

2. Створення двух кнопок “Вибрати файл” та “Перевірити” та їх місце розташування.

```
<HBox alignment="CENTER" prefHeight="71.0" prefWidth="807.0">
        <children>
          <Button fx:id="selectFileButton" mnemonicParsing="false" onMouseClicked="#selectFile" text="Вибрати файл">
            <font>
              <Font size="18.0" />
            </font>
          </Button>
          <Separator prefHeight="2.0" prefWidth="90.0" visible="false" />
          <Button fx:id="checkHashButton" disable="true" mnemonicParsing="false" onMouseClicked="#checkHash" text="Перевірити">
            <font>
              <Font size="18.0" />
            </font>
          </Button>
```

```
    </children>
  </HBox>
```

Далі було створено контролер HashCheckFormController.java

Приклад створення методів у HashCheckFormController.java:

1. Метод ініціалізації

```
public void initialize() {}
```

2. Метод, який викликається при натисканні кнопки “Перевірити”

```
void checkHash(MouseEvent event) {}
```

3. Метод, який викликається при натисканні кнопки “Вибрати файл”

```
void selectFile(MouseEvent event) {}
```

Розглянемо детально один із двух методів для логіки роботи програми:

1. Метод перевірки результатів

```
void checkHash(MouseEvent event) {
    try {
        String str = textArea.getText();

        int startIndex = str.lastIndexOf(":");
        String hash = str.substring(startIndex + 2);

        int endIndex = str.lastIndexOf("\n");
        String data = str.substring(0, endIndex);

        String calculatedHash = Utils.toHexString(Utils.getHash(data));

        if (calculatedHash.equals(hash)) {
            Alert alert = new Alert(Alert.AlertType.INFORMATION, "Результат вірний!", ButtonType.OK);
```

```
                alert.setTitle("OK");
                alert.showAndWait();
            } else {
                Alert alert = new Alert(Alert.AlertType.ERROR, "Результат було змінено!", ButtonType.OK);
                alert.setTitle("Помилка");
                alert.showAndWait();
```

4.2 Опис програми

Після запуску програми для студента, з’являється початкове вікно, де розташована інформація про автора та дві кнопки з різними режимами “Режим тренінгу” та “Режим оцінювання” рис.(4.3). В даній роботі буде розглядатися режим оцінювання, детально про режим тренінгу можна дізнатися в бакалаврській роботі [2].

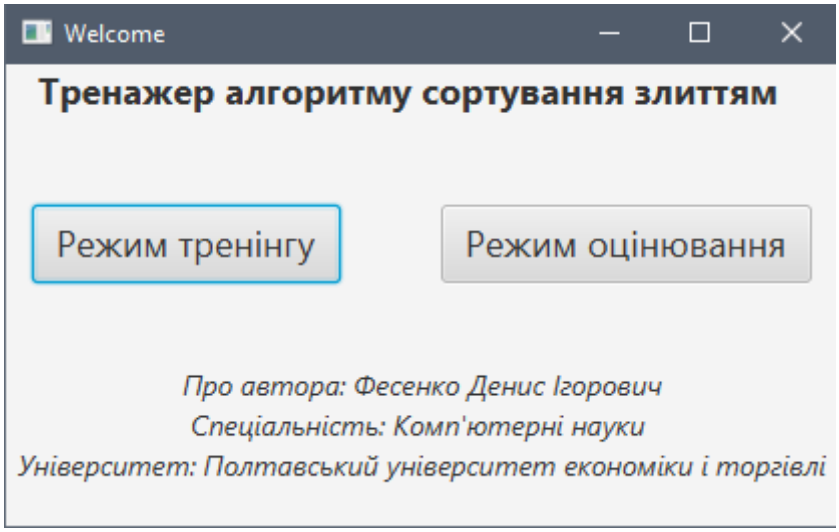

Рисунок 4.3 – Початкове вікно з різними режимами

Після натискання кнопки “Режим оцінювання” відкриється вікно, де користувачу потрібно ввести особисті дані рис. (4.4).

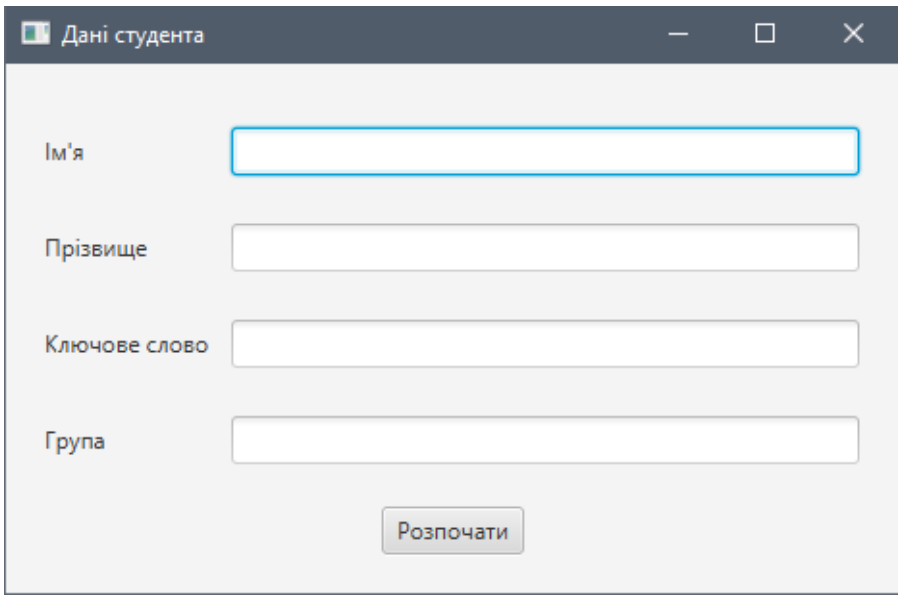

Рисунок 4.4 – Вікно для особистих даних

Щоб почати трасування алгоритму, потрібно натиснути кнопку “Розпочати” після чого з’явиться вікно де розташований алгоритм, масив чисел, поля для відповідей та лог-файл рис. (4.5).

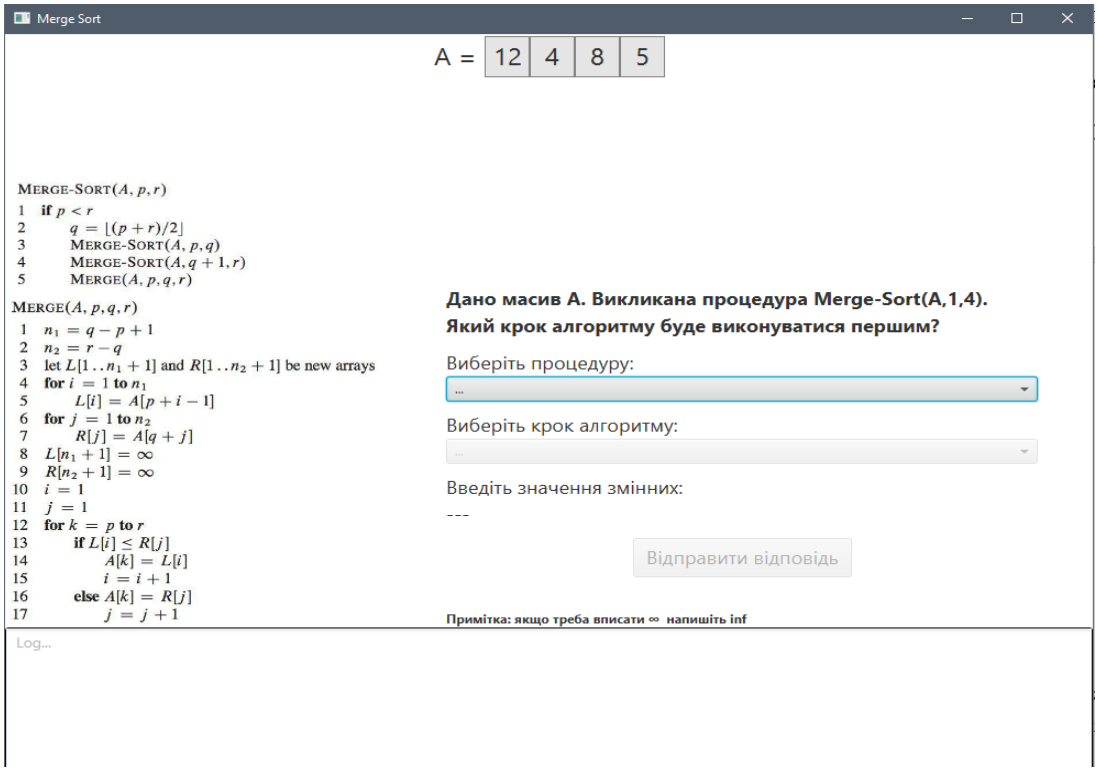

Рисунок 4.5 – Вікно з алгоритмом та прикладом

Далі студент зможе вибрати яка процедура та крок алгоритму буде виконуватися, після натиснення на поле “Виберіть процедуру” з’явиться список усіх операцій, аналогічно с полем “Виберіть крок алгоритму”, рис. (4.6-4.8).

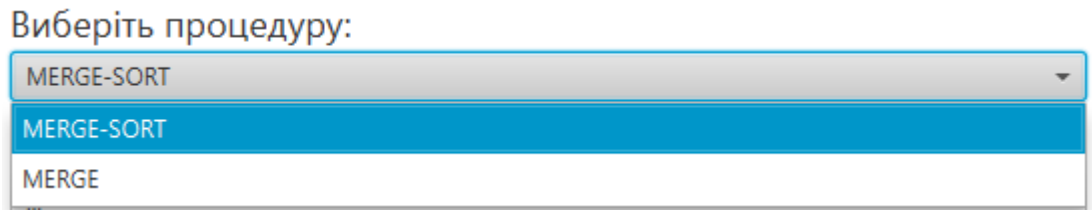

Рисунок 4.6 – Список поля “Виберіть процедуру”

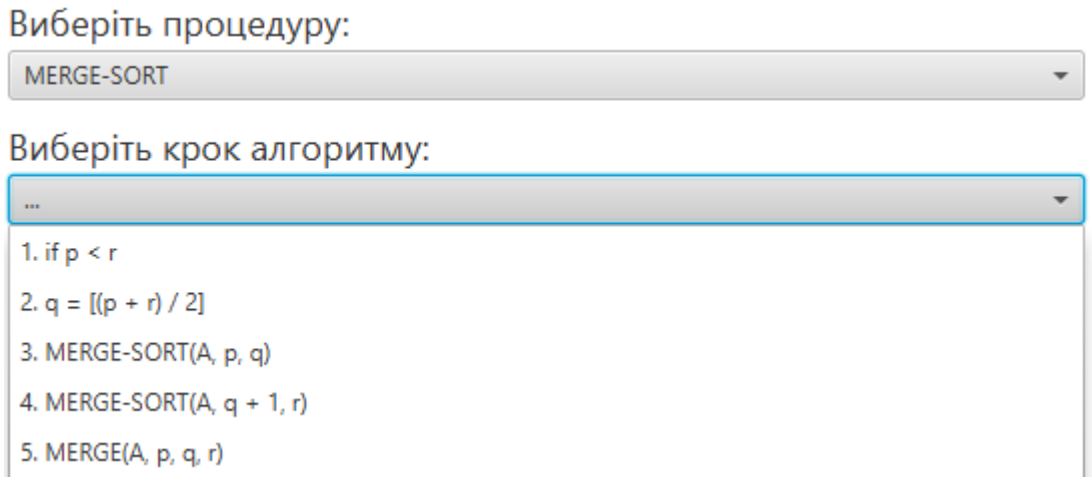

Рисунок 4.7 – Список поля “Виберіть крок алгоритму” процедури Merge-Sort

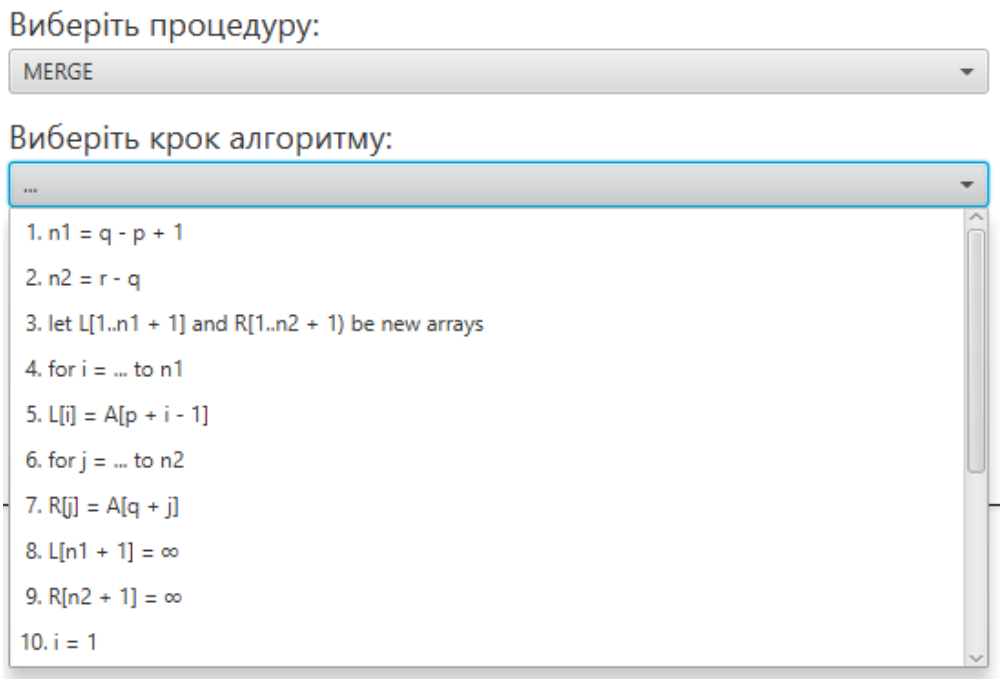

Рисунок 4.8 - Список поля “Виберіть крок алгоритму” процедури Merge

Після вибору процедури та кроку з’явиться поля куди можна ввести відповідь та кнопка для відправки, рис. (4.9).

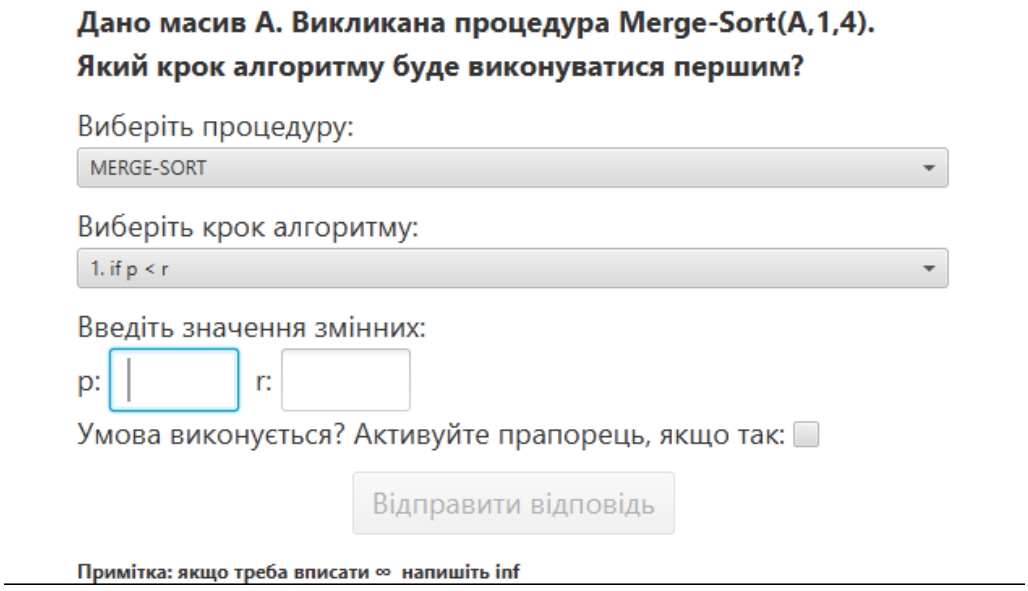

Рисунок 4.9 – Поля для відповіді та кнопка для відправки

Якщо користувач вводить відповідь неправильно, як на рис. (4.10), з’являється повідомлення про те, що відповідь була не вірна, якщо користувач вводить відповідь не правильно багато разів з’явиться повідомлення з підказкою, як на рис. (4.11), також якщо користувач введе не цифрові значення або не виняткове слово, з’явиться повідомлення про введення не цифрових даних, рис. (4.12).

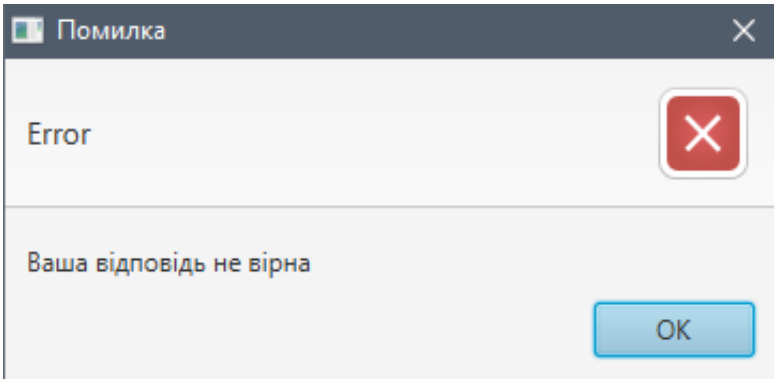

Рисунок 4.10 – Повідомлення про неправильну відповідь

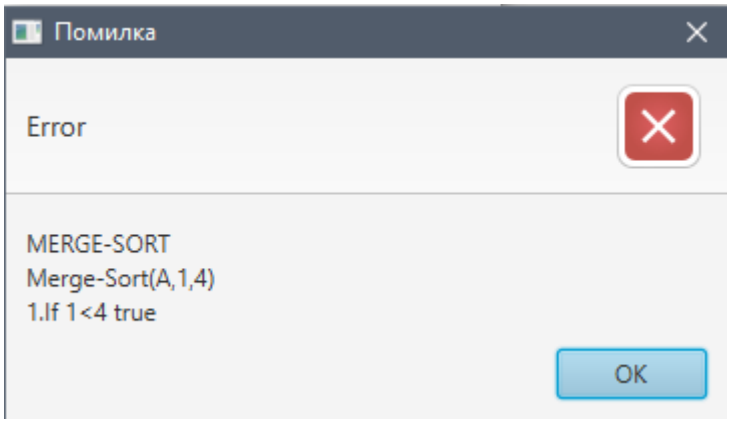

Рисунок 4.11 – Повідомлення з підказкою

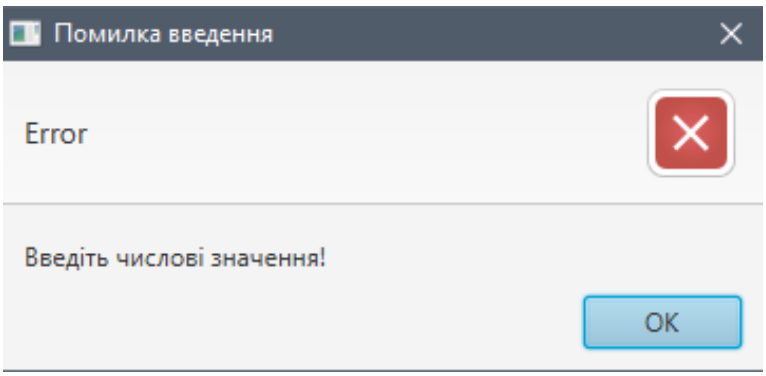

Рисунок 4.12 - повідомлення про введення не цифрових даних

Коли користувач вводить правильну відповідь, його відповідь зберігається в лог-файлі, як на рис. (4.13).

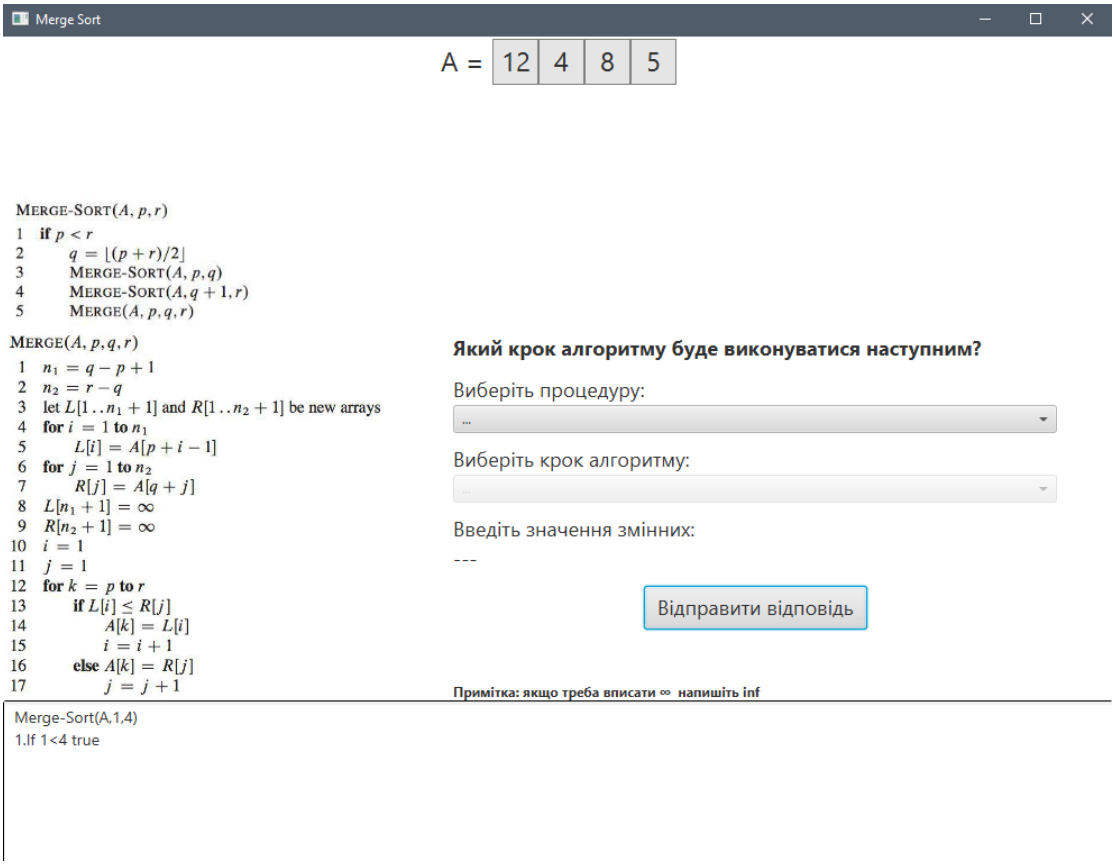

Рисунок 4.13 – Лог-файл

Після закінчення прикладу, студент отримає тестові запитання з варіантами відповідей рис. (4.14).

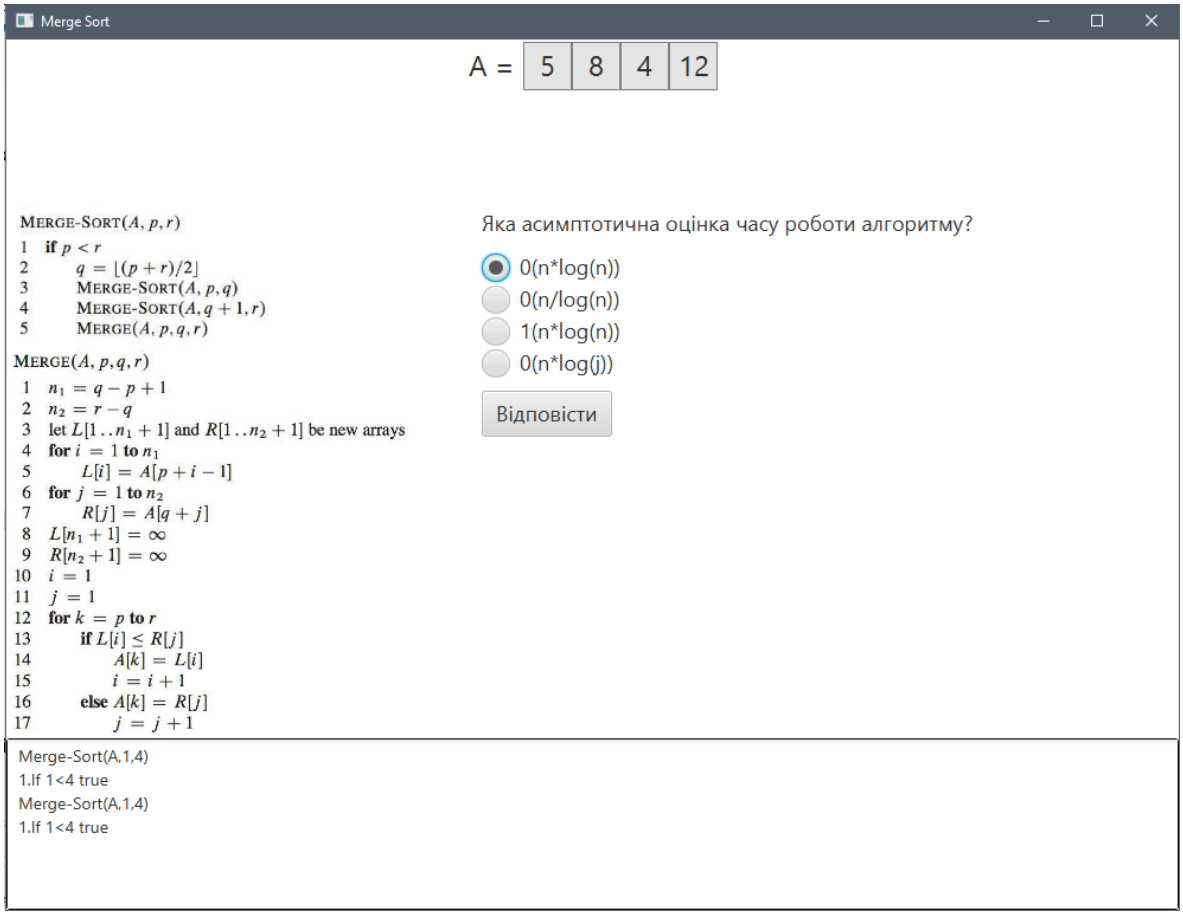

Рисунок 4.13 – Тестові запитання

Якщо користувач вводить відповідь неправильно, як на рис. (4.14), з’являється повідомлення про те, що відповідь була не вірна, якщо користувач вводить відповідь не правильно багато разів з’явиться повідомлення з підказкою, як на рис. (4.15).

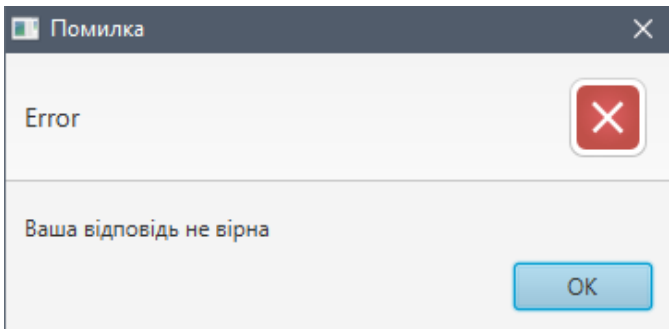

Рисунок 4.14 – Повідомлення про неправильну відповідь

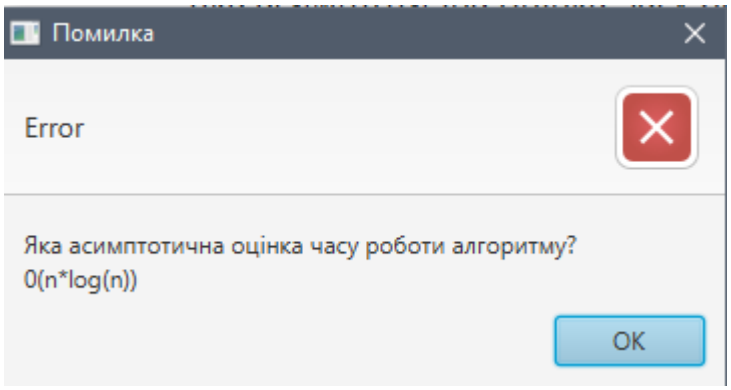

Рисунок 4.15 – Повідомлення з підказкою

Після проходження тестових запитань, з’являється вікно з детальною інформацією про успішність проходження тренінгу рис. (4.16).

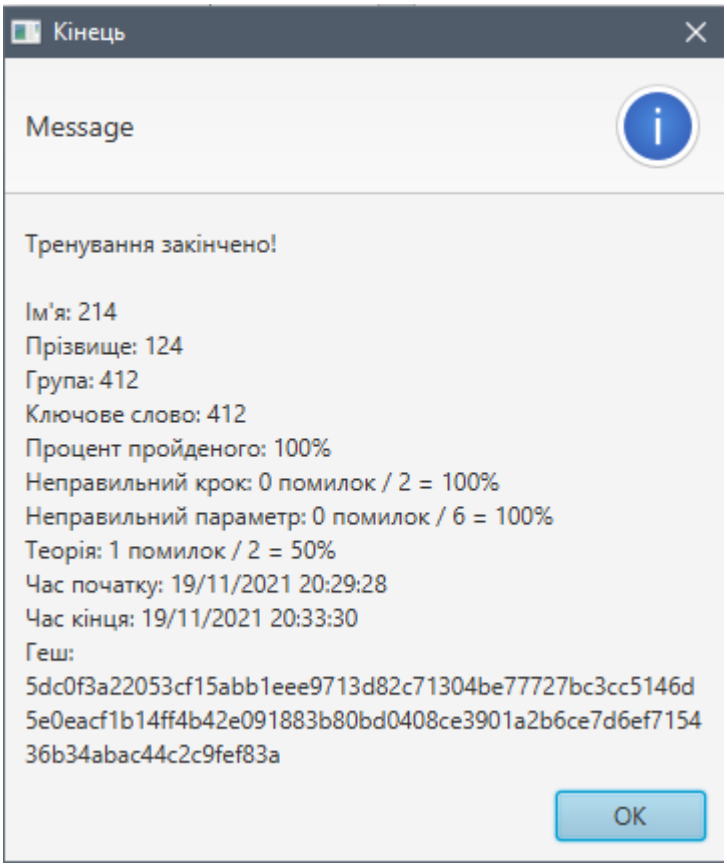

Рисунок 4.16 – Вікно с детальною інформацією

Вся детальна інформація після проходження прикладу зберігається у файл для відправки викладачу рис. (4.17).

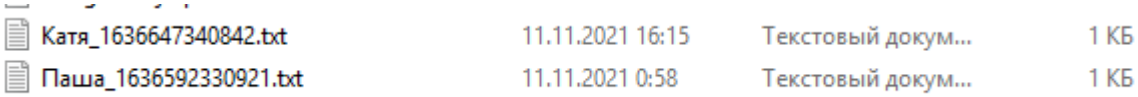

Рисунок 4.17 – Файли для відправки викладачу

Після запуску програми для викладача, з’являється початкове вікно, де розташована поле та дві кнопки “Вибрати файл” і “Перевірити” рис. (4.18).

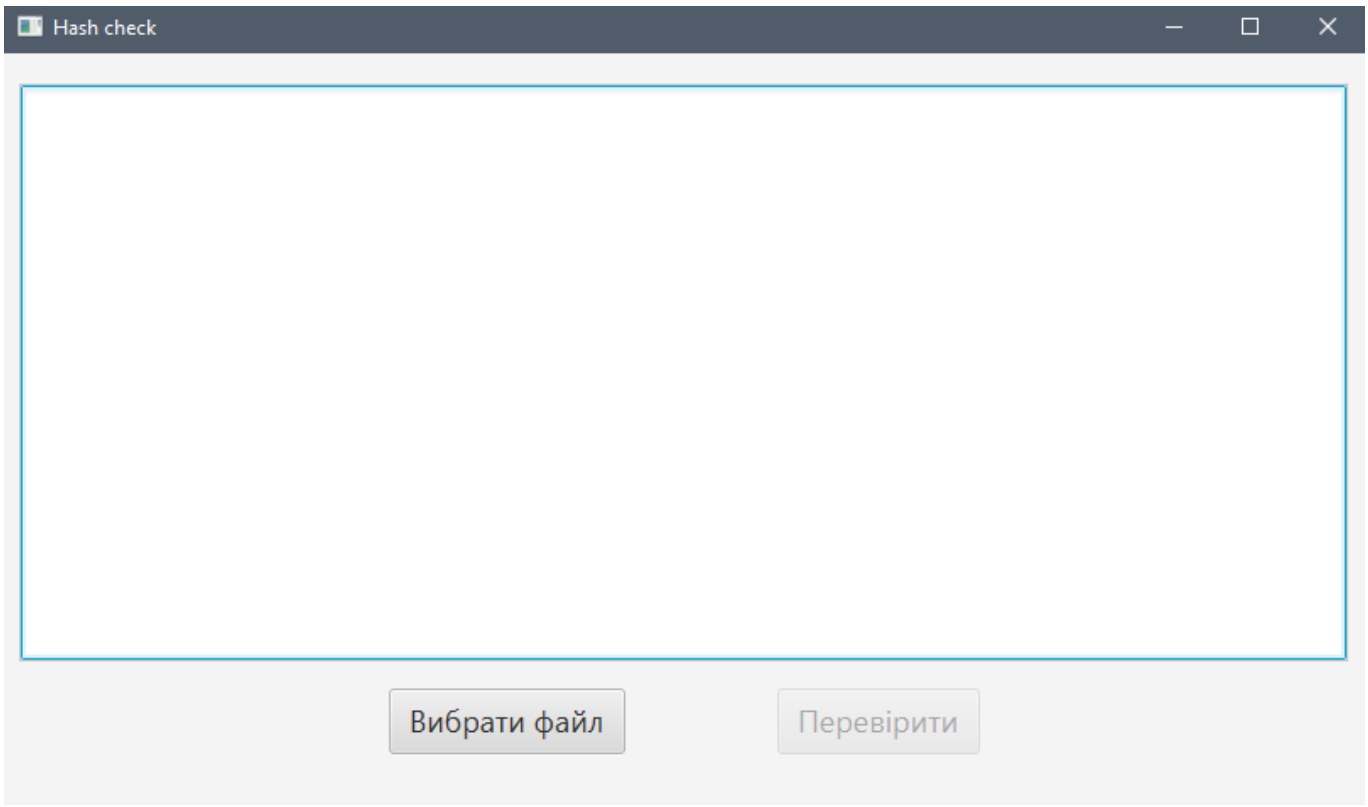

Рисунок 4.18 – Початкове вікно

Після натискання на кнопку “Вибрати файл”, можна буде обрати файл для перевірки та після вибору файлу, його інформація з’явиться у полі рис.(4.19-4.20).

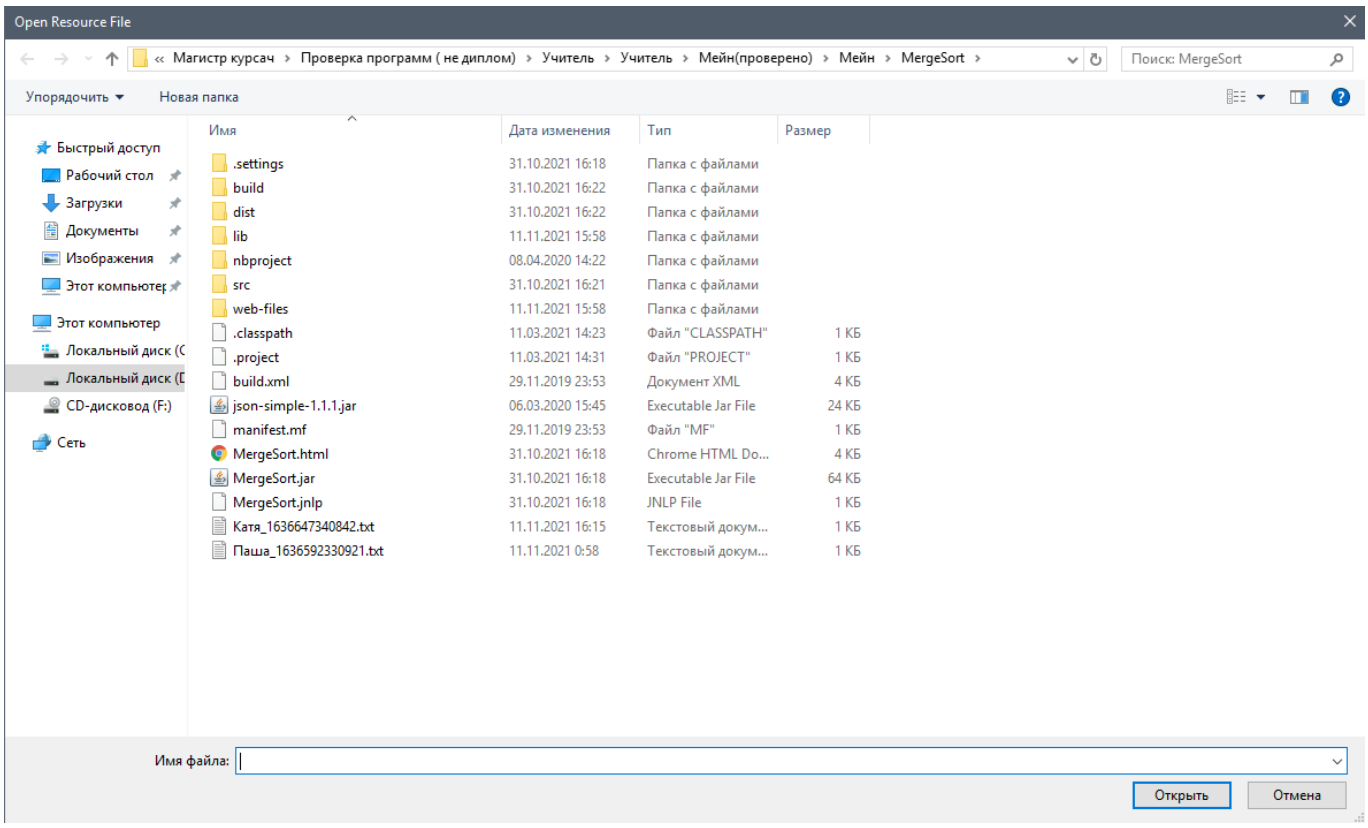

Рисунок 4.19 – Вікно для вибору файлу для перевірки

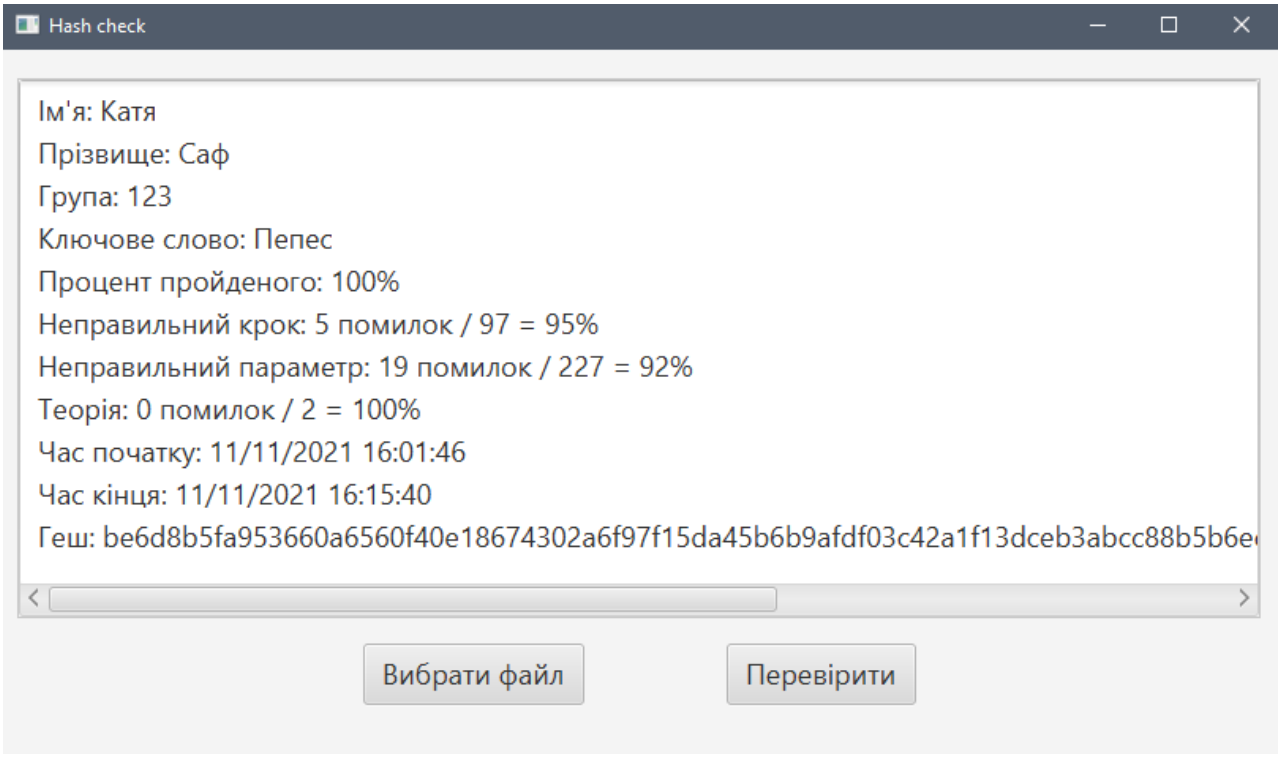

Рисунок 4.20 – Поле з інформацією

Після натискання на кнопку “Перевірити”, програма повідомить викладача про те чи були результати змінені чи ні рис.(4.21-4.22).

Рисунок 4.21 – Повідомлення про правдивість результатів

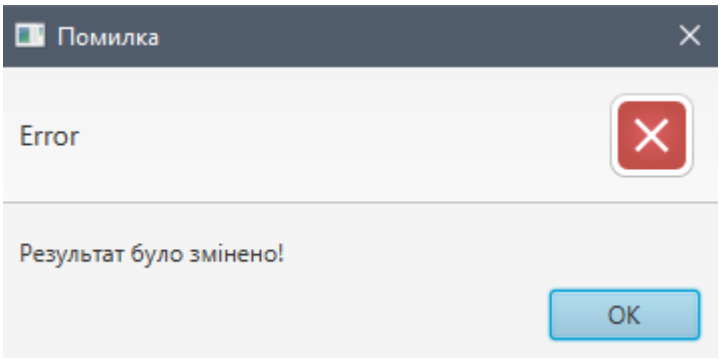

Рисунок 4.22 – Повідомлення про зміну результатів

4.3 Перевірка валідності

Було здійснено перевірку на валідність програмного комплексу, що дає змогу зрозуміти чи виконує функції які було поставлено у меті та чи повністю він працездатний.

Для цього було проведено тестування програмного комплексу.

1. Введення недопустимих символів, як на рис. (4.23).

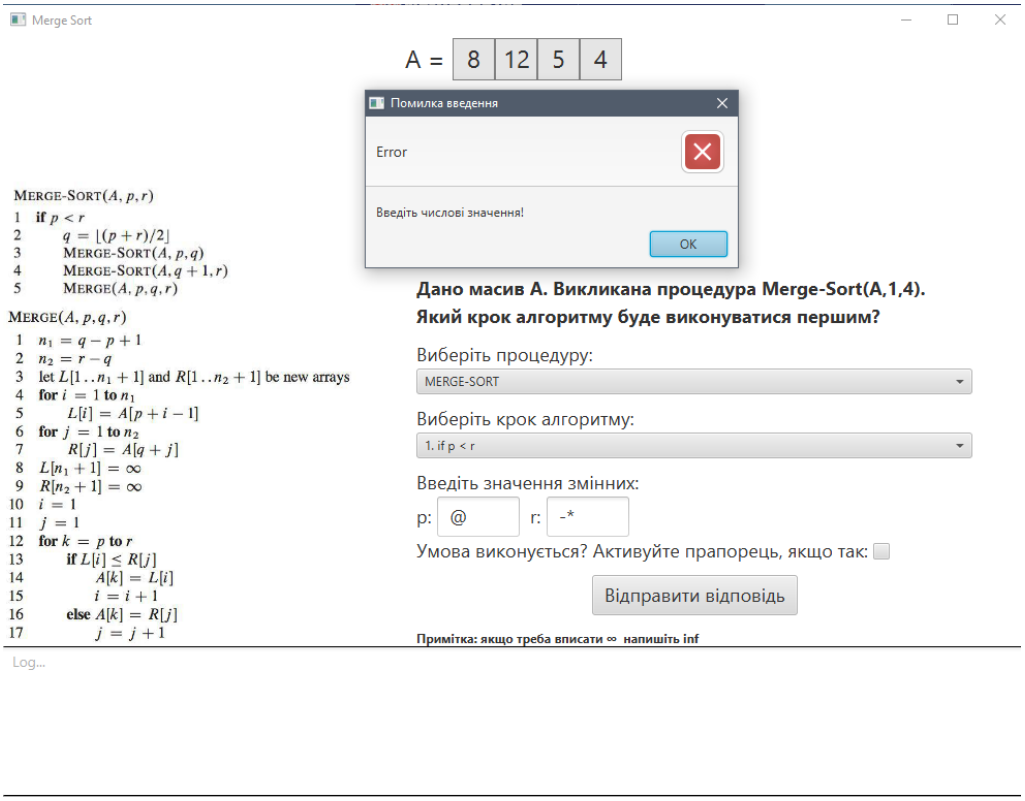

Рисунок 4.23 – Недопустимі символи

2. Вибір не правильної процедури або кроку алгоритму, як на рис. (4.24).

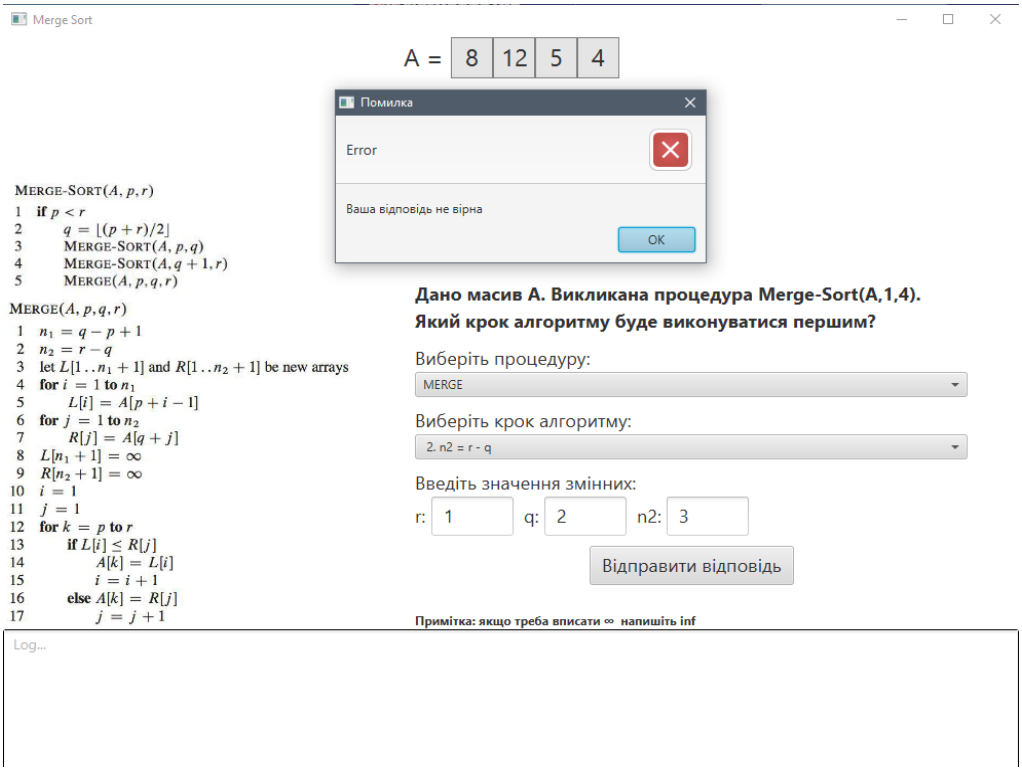

Рисунок 4.24 – Не правильний вибір процедури або кроку алгоритму

3. Вибір правильної процедури або кроку алгоритму та введення правильної відповіді, як на рис. (4.25-4.26).

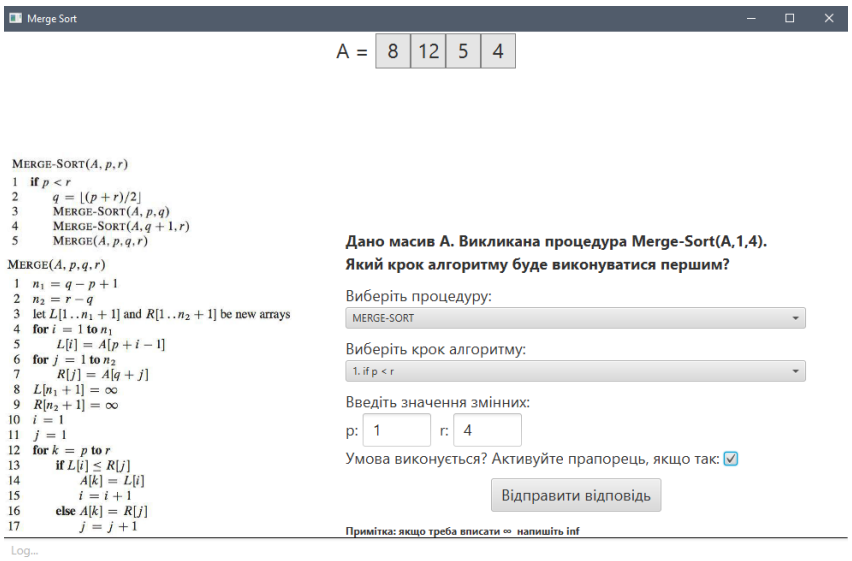

Рисунок 4.25 - Вибір правильної процедури або кроку алгоритму та введення правильної відповіді

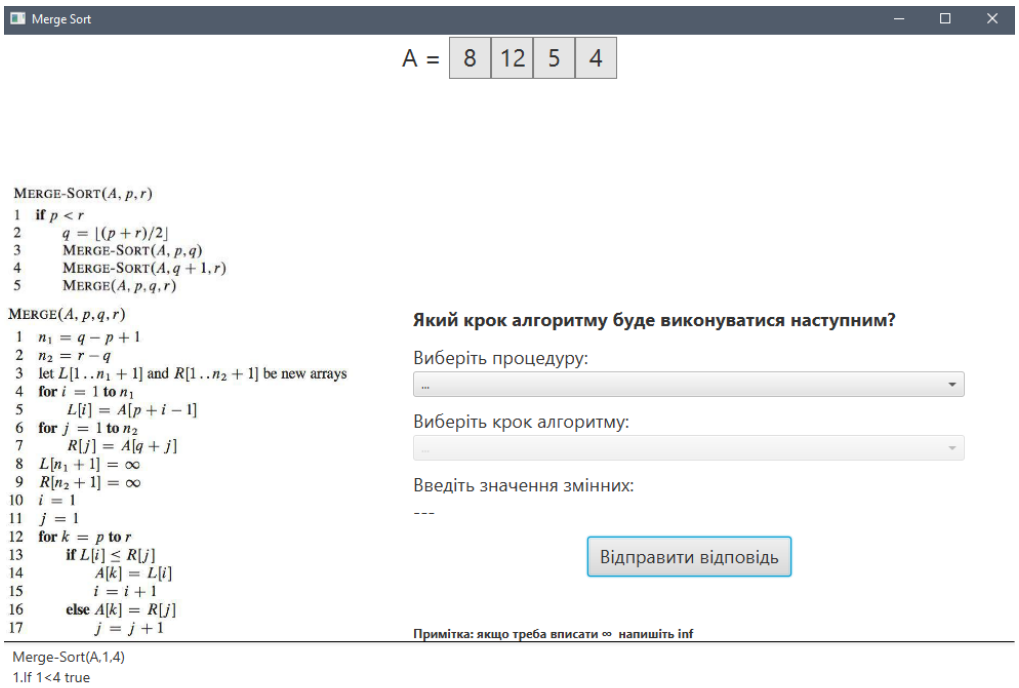

Рисунок 4.26 – Занесення правильної відповіді у лог-файл

4. Вибір файлу з змінними результатами рис. (4.27).

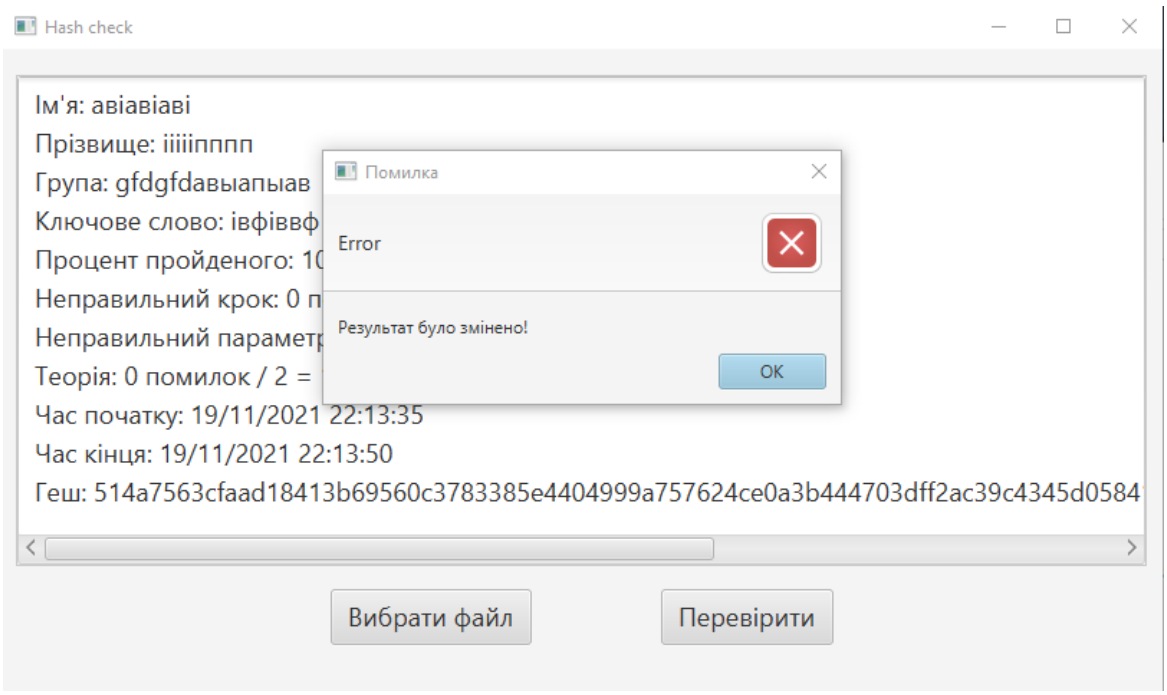

Рисунок 4.27 – Файл з змінними результатами

5. Вибір файлу з правдивими результатами рис. (4.28).

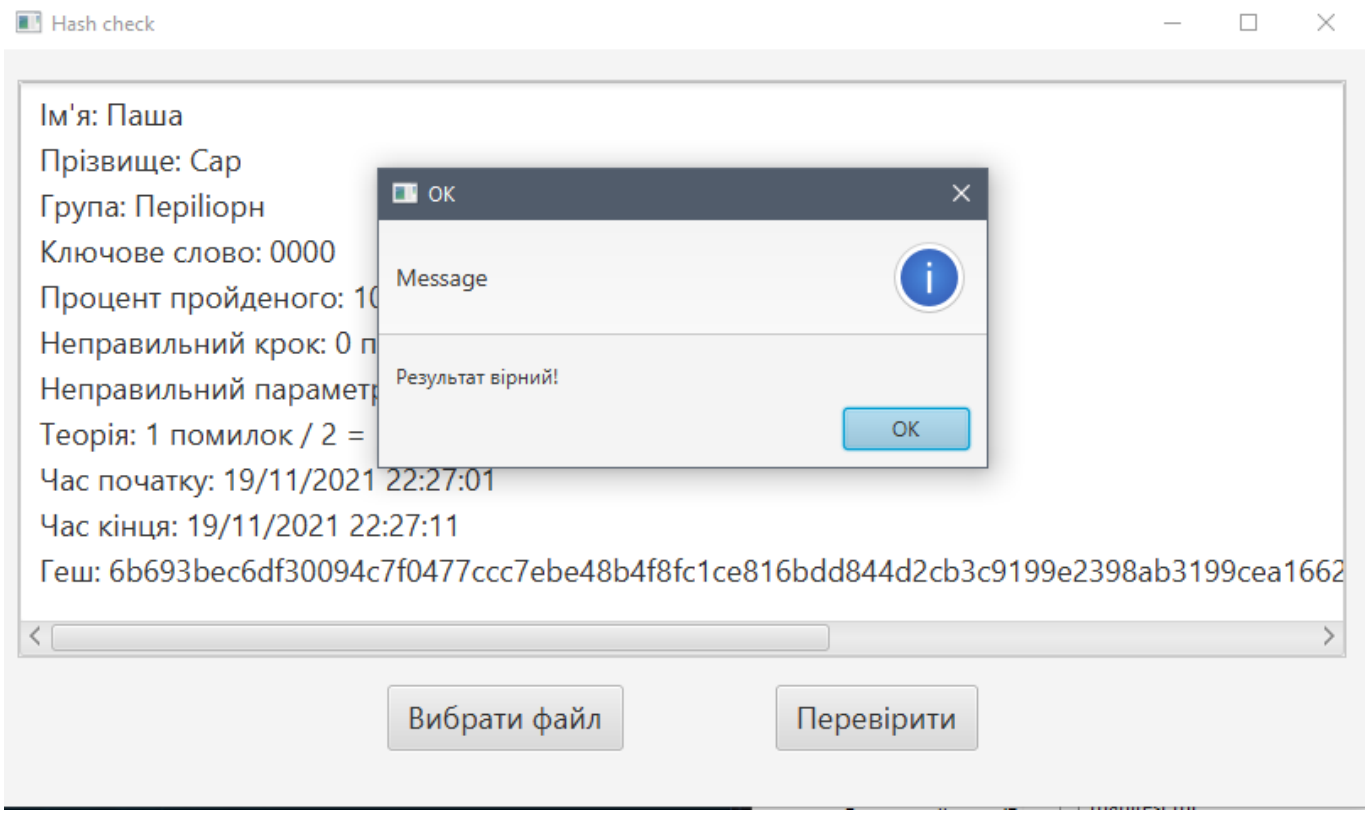

Рисунок 4.28 – Файл з правдивими результатами

Валідність програмного комплексу показує , що він перевіряє тільки ті дані, які повинен перевіряти.

4.4 Необхідна користувачу програмна інструкція

Для роботи с програмним комплексом, необхідно встановити JDK 1.8. При запуску програми для студента з’являється вікно, де розташовано дві кнопки “Режим тренінгу” та “Режим оцінювання”. Після натискання на кнопку “Режим оцінювання” з’являється вікно, де користувачу потрібно ввести свої персональні дані. Далі відкриється основній інтерфейс, зверху буде розташований випадковий масив чисел, зліва знаходиться алгоритм, за яким користувач повинен розв’язати приклад. Справа знаходяться два поля, в одному потрібно вибрати процедуру Merge-Sort чи Merge, у другому потрібно вибрати крок зі списку. Далі з’являться поля, де потрібно вписати відповідь у цифровому форматі чи винятковим символом

та натиснути кнопку “Відправити відповідь”. Якщо студент правильно відповів то його відповідь запишеться вниз у лог-файл, де будуть вноситись усі подальші правильні відповіді поки тренінг не закінчиться. Після закінчення з’являться повідомлення з детальною інформацією про успішність та вся інформація збережеться у файл.

При запуску програми для викладача з’являється вікно з полем та дві кнопки “Вибрати файл” і “Перевірити”. Після натиснення кнопки “Вибрати файл”, потрібно вибрати файл, який скинув користувач. Коли файл обрано, у полі з’явиться інформація про успішність проходження тренінгу. Після натиснення кнопки “Перевірити” програма перевірить файл та з’явиться повідомлення, в якому буде зрозуміло чи було змінено результати чи ні.

ВИСНОВКИ

При виконанні магістерської роботи було створено програмний комплекс для виконання та оцінювання завдань.

Створений програмний комплекс може бути впроваджено в дистанційне навчання ПУЕТ в дистанційний курс «Аналіз алгоритмів» студентів спеціальності «Комп’ютерні науки».

Виконані основні вимоги для створення програмного комплексу, а саме:

1. Складено алгоритм для двох програм.
2. Складено блок-схеми відповідно до обраних алгоритмів.
3. Програмно реалізовано дві програми.
4. Перевірено працездатність усього програмного комплексу.

Виконані основні вимоги до роботи та використання програмного комплексу:

1. Інтерфейс програм зрозумілий для студента та викладача.
2. Реалізовано введення персональних даних.
3. При кожному введені відповіді, реалізовано перевірку даних та проінформувати студента, якщо відповідь неправильна та надати змогу відповісти ще раз та надати підказку, якщо студент багато раз відповів не правильно.
4. Реалізовано відстежування усіх попередніх кроків у лог-файл.
5. Після проходження прикладу отримати детальну інформацію про успішність проходження та зберігання їх у файл.
6. Реалізована швидка та якісна перевірка файлу та в разі зміни результатів повідомити про це.

Програмний комплекс дотримав усіх вимог щодо розробки та роботи, шо означає надійне функціонування та ефективність його у використані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Белінська В.В. Створення програмного забезпечення тренажера з теми «Розподіли дискретних випадкових величин та їх числові характеристики» дистанційного навчального курсу «Теорія ймовірностей та математична статистика / В. В. Белінська, Т. О. Парфьонова – Полтава: Кафедра ММСІ ПУЕТ, 2020. – 63 с. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/10305

2. Фесенко Д. І. Розробка тренажеру з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів» / Д.І. Фесенко, – Полтава: Кафедра ММСІ ПУЕТ, 2020. – 38 с. – Режим доступу: http://dspace.puet.edu.ua/bitstream/123456789/9001/1/Fesenko_Denis_Igorovich-KNIT-41%20.pdf

3. Жайворонок Я.І. Розробка програмного забезпечення для тренажера дистанційного навчального курсу «Теорія інформації та кодування» з теми «Коди із виявленням помилок» / Я.І. Жайворонок – Полтава: Кафедра ММСІ ПУЕТ, 2020. – 113 с. – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/10045

4. Глущенко К.С., Гавриленко О.В. Аналіз методів порівняння текстів у випускних роботах студентів [Електронний ресурс]: Fiot. – Режим доступу: http://fiot.kpi.ua/wp-content/uploads/2016/06/%D1%82%D0%B5%D0%B7%D0%B8_%D0%90%D0%A1%D0%9E%D0%86%D0%A3_%D0%86%D0%9E%D0%A2_2016.pdf

5. Ємець О. О. Про розробку тренажерів для дистанційних курсів кафедрою ММСІ ПУЕТ / О.О. Ємець // Інформатика та системні науки (ІСН-2015): матеріали VI Всеукраїнської науково-практичної конференції за міжнародною участю, (м. Полтава, 19–21 берез. 2015 р.). – Полтава: ПУЕТ, 2015. – С. 152-161 – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/2488

6. Парфьонова Т. О. Про розробку тренажерів для дистанційного навчального курсу "Алгебра і геометрія" / Т. О. Парфьонова // Інформатика та системні науки (ІСН-2016): матеріали VII Всеукраїнської науково-практичної конференції за міжнародною участю, (м. Полтава, 10–12 берез. 2016 р.) – Полтава: ПУЕТ, 2016 – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/2974

7. Чілікіна Т. В. Огляд тренажерів з дисципліни "Математичний аналіз" на прикладі розробок студентів напряму "Інформатика" / Т. В. Чілікіна // Інформатика та системні науки (ІСН-2016): матеріали VII Всеукраїнської науково-практичної конференції за міжнародною участю, (м. Полтава, 10–12 берез. 2016 р.). – Полтава: ПУЕТ, 2016. – С. 329-330 – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/2993?mode=full

8. Кормен Т. Алгоритмы: построение и анализ, 2-е изд./ Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн — М.: Вильямс, 2005. — 1296 с – Режим доступу: https://e-maxx.ru/bookz/files/cormen.pdf

9. JavaFX [Електронний ресурс]: Matanit. – Режим доступу: https://metanit.com/java/javafx/2.1.php

10. Герберт Шилдт. Java: Полное руководство, 9-е издание/Ш.Герберт, 2015 [Електронний ресурс]: Library. – Режим доступу: https://library-it.com/programming/java/java-8-polnoe-rukovodstvo-9-e-izdanie-2015/

11. Сортування злиттям [Електронний ресурс]: Tutorialspoint. - Режим доступу: https://www.tutorialspoint.com/data_structures_algorithms/merge_sort_algorithm.htm

12. Хеш-функція [Електронний ресурс]: Academic. – Режим доступу: https://dic.academic.ru/dic.nsf/ruwiki/746554

13. Криптографія [Електронний ресурс]: Habr. – Режим доступу: https://habr.com/ru/post/444974/

14. Пасєка М.С. Аналіз використання високоефективних реалізації функції хешування SHA-512 для розробки програмних систем [Електронний ресурс]: Lib. – Режим доступу: http://lib.pnu.edu.ua:8080/bitstream/123456789/2243/1/68-%D0%A2%D0%B5%D0%BA%D1%81%D1%82%20%D1%81%D1%82%D0%B0%D1%82%D1%82%D1%96-300-1-10-20190506.pdf

15. Фесенко Д. І. Програмний комплекс для виконання та оцінювання завдань для студентів та вчителів денної або дистанційної форми з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів». / Д.І. Фесенко, Ю. Ф. Олексійчук – Полтава: Кафедра ММСІ ПУЕТ, 2021. – 4 с – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/10328

16. Фесенко Д. І. Розробка тренажеру з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів» / Д.І. Фесенко, Ю.Ф Олексійчук – Полтава: Кафедра ММСІ ПУЕТ, 2020. – 4 с – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/8290

17. Ярмоленко А. В. Алгоритм роботи тренажеру з теми «Асимптотичні оцінки функцій» дисципліни «Аналіз алгоритмів» / А. В. Ярмоленко, Ю. Ф. Олексійчук // Комп'ютерні науки і прикладна математика (КНіПМ-2018): матеріали науково-практичного семінару. Випуск 2 – Полтава: Кафедра ММСІ ПУЕТ, 2018. – С.14-16 – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/6974

18. Голубенко Вл. О. Програмна реалізація тренажеру з теми «Сортування бульбашками» дисципліни «Аналіз алгоритмів» / В. О. Голубенко, Ю. Ф. Олексійчук // Комп'ютерні науки і прикладна математика (КНіПМ-2018): матеріали науково-практичного семінару. Випуск 2 – Полтава: Кафедра ММСІ ПУЕТ, 2018. – С. 6-9 – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/6976

19. Русін В. С. Програмна реалізація елементів тренажеру з теми "Аналіз алгоритму сортування вставками" дисципліни "Аналіз алгоритмів" / В. С. Русін, Ю. Ф. Олексійчук // Інформатика та системні науки (ІСН-2017): матеріали VIII Всеукраїнської науково-практичної конференції за міжнародною участю (м. Полтава, 16–18 березня 2017 р.) – Полтава: ПУЕТ, 2017. – С. 236-237 – Режим доступу: http://dspace.puet.edu.ua/handle/123456789/5654

20. Хешування [Електронний ресурс]: Kaspersky. – Режим доступу: https://www.kaspersky.ru/blog/the-wonders-of-hashing/3633/