ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ
ФОРМА НАВЧАННЯ ДЕННА
КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту

Завідувач кафедри _____О.О. Ємець

(підпис)

«_____»_____2020 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему

Розробка тренажеру з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів»

зі спеціальності 122 «Комп'ютерні науки та інформаційні технології»

Виконавець роботи Фесенко Денис Ігорович

_____ «___»_____2020 р.

(підпис)

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

_____ «___»_____2020 р.

(підпис)

ПОЛТАВА 2020 р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, символів, скорочень
Лог-файл	Місце де зберігається інформація про події у системі
Merge-Sort	Алгоритм сортування
Merge	Злиття
Парсинг	Синтаксичний аналіз

Вступ

Актуальність даної роботи полягає в створені тренажеру для студентів денної або дистанційної форми з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів».

Один з важливих пунктів актуальності використання тренажеру є те, що він допомагає користувачу краще розібратися з роботою алгоритму.

Також наявність тренажеру надає допомогу не тільки студенту, а й викладачеві тому, що:

1) студент може самостійно провести трасування алгоритму без присутності викладача;

2) тренажер вказує на допущені помилки та дає підказки.

3) після завершення трасування алгоритму, студент має змогу переглянути лог-файл знизу тренажера , щоб потім зробити аналіз його складності.

Мета даної роботи – створення тренажеру, який допомагає користувачу провести трасування алгоритму, щоб краще зрозуміти його роботу та для вивчення основних принципів алгоритму сортування злиттям з теми «Аналіз алгоритму сортування злиттям».

Для досягнення мети були поставленні такі завдання:

- вивчення методичних рекомендацій та стандартів для оформлення та виконання бакалаврських робіт;
- ознайомлення з літературою, яка стосується алгоритму сортування злиттям;
- інформаційний огляд з даної теми;
- складання блок-схеми алгоритму тренажеру;
- програмування навчального-тренажеру;
- тестування всього тренажеру.

Об’єктом розробки є програмування тренажерів для дистанційного курсу «Аналіз алгоритмів».

Предметом розробки навчальний тренажер з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів».

Методи розробки. Для створення алгоритму використано алгоритм сортування злиттям. Для створення тренажеру застосовано об’єктно-орієнтовану мову програмування Java із застосуванням вільного інтегрованого середовища розробки NetBeans.

Новизною даної роботи є тренажер з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів». При пошуку в інтернеті не було знайдено жодного тренажеру на дану тему.

Практичною цінністю тренажеру є створення якісного способу для навчання студента алгоритму сортування злиттям.

Розроблений тренажер 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: 38 сторінки, 24 рисунків, 1 додаток (на 30 сторінках) та 10 літературних джерел.

1 ПОСТАНОВКА ЗАДАЧІ

1.1 Постановка задачі розробки тренажера

Задачею даної роботи є розробка навчального тренажера для студентів з теми «Аналіз алгоритму сортування злиттям» дисципліни «Аналіз алгоритмів».

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

Розглянути основні вимоги до роботи та використання тренажеру.

Основними вимогами для розробки навчального тренажеру є:

1. Скласти алгоритм навчального тренажеру.
2. Скласти блок-схему відповідно до розробленого алгоритму.
3. Програмно реалізувати навчальний тренажер.
4. Перевірити працездатність усього тренажеру.

Основними вимогами до роботи та використання навчального тренажеру студентом є:

1. Інтерфейс програми повинен бути інтуїтивно зрозумілим студенту.
2. При кожному введені відповіді, реалізувати перевірку даних та проінформувати студента, якщо відповідь неправильна та надати змогу відповісти ще раз та надати підказку, якщо студент багато раз відповів не правильно.
3. Після завершення проходження прикладу дати змогу студенту подивиться всі кроки, які він відповів та дати змогу скопіювати їх собі.

2 ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1 Огляд робіт, де розглянуте аналогічне до теми завдання

Під час аналізу схожих тренажерів, були розглянуті тренажери з дистанційного курсу «Аналіз алгоритмів» студентів ПУЕТ попередніх років, а саме тренажер Ярмоленко А. В. [1], Голубенко Вл. О. [2], Русін В. С. [3]. Всі розглянуті навчальні тренажери порівнювалися з висунутими вимогами.

Тренажер Ярмоленко А. В. [1] з теми «Асимптотичні оцінки функцій» дисципліни «Аналіз алгоритмів». Тренажер виконаний на об’єктно-орієнтованій мові Java. Перед початком тренінгу виводиться інформація про тренажер далі випадково вибираються кілька завдань.

Тренажер Голубенко Вл. О. [2] з теми «Сортування бульбашками» дисципліни «Аналіз алгоритмів». Тренажер виконаний на об’єктно-орієнтованій мові Java. При запуску тренажера відображується вікно, де наведена інформація про тренажер, присутня можливість ознайомитись з навчальним матеріалом та розпочати тренінг. Тренінг включає в себе тестові запитання.

Тренажер Русін В. С. [3] з теми "Аналіз алгоритму сортування вставками" дисципліни "Аналіз алгоритмів". Тренажер виконаний на об’єктно-орієнтованій мові JavaScript для реалізації веб-тренажеру. При створенні графічного інтерфейсу була використана мова HTML та каскадні таблиці стилів CSS. При запуску тренажера з’являється поле з заголовком, код алгоритму, поле з масивом та історія дій. Тренінг включає в себе випадковий масив.

2.2 Позитивні аспекти оглянутих робіт

Переваги тренажера Ярмоленко А. В. [1]:

1. Велика кількість питання, що дає змогу добре закріпити знання.
2. Присутня інформація про помилку та підказку.

Переваги тренажера Голубенко Вл. О. [2]:

1. Є можливість ознайомитись з навчальним матеріалом.
2. В питаннях з розрахунками, студенту показані формули.

Переваги тренажера Русін В. С. [3]:

1. Наявність історії дій.
2. Показано код алгоритму сортування.
3. Використано технологію адаптивного дизайну за допомогою якого можна відкрити тренажер не тільки на комп’ютері.
4. Приклади створюються автоматично що дає змогу практикуватися на нових прикладах без повторень.

2.3 Вади розробок з оглянутих робіт

Недоліки тренажеру Ярмоленко А. В. [1]:

1. Відсутні будь-які теоретичні відомості.
2. Не завжди зрозуміло написанні відповіді до питань.
3. Тільки тестові питання без введення власної відповіді.

Недоліки тренажеру Голубенко Вл. О. [2]:

1. Відсутність історії дій
2. При повторному проходженні тренажера завжди будуть всі ті питання, які були раніше.

Недоліки тренажеру Русін В. С. [3]:

1. Не має можливість перевірити чи правильно тренажер перевіряє відповіді так, як тренажер автоматично створює приклади а не використовує вбудовані та перевірені приклади.

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

Алгоритму сортування злиттям використовує мало додаткової пам’яті, але має квадратичну складність. 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 (зливаються) в правильному порядку.

Приклад:

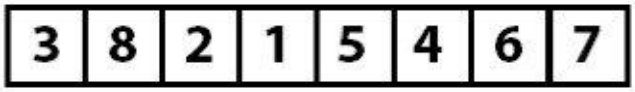

Рисунок 1.1 – Приклад

Ділимо його навпіл:

Рисунок 1.2 – Поділ на дві групи

І ділимо кожну частину навпіл, поки не залишаться частини з одним елементом:

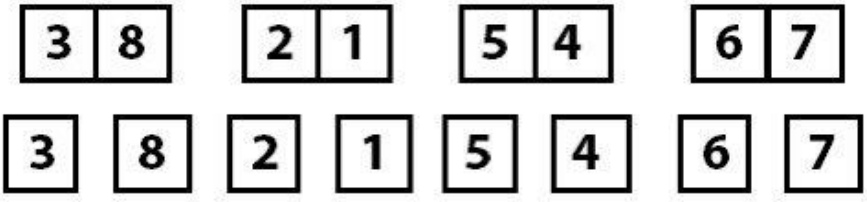

Рисунок 1.3 – Поділ на групи з одним елементом

Тепер, коли масив поділений на максимально короткі ділянки, зливаємо їх у правильному порядку:

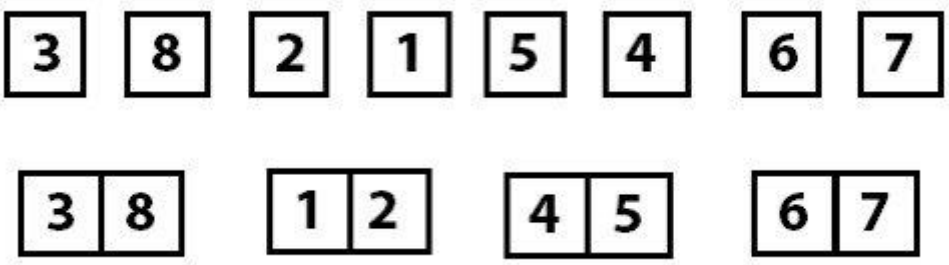

Рисунок 1.4 – Зливаємо групи у правильному порядку

Спочатку отримуємо групи по два відсортовані елементи, потім “збираємо” їх у групи по чотири елементи і наприкінці збираємо всі разом у відсортований масив.

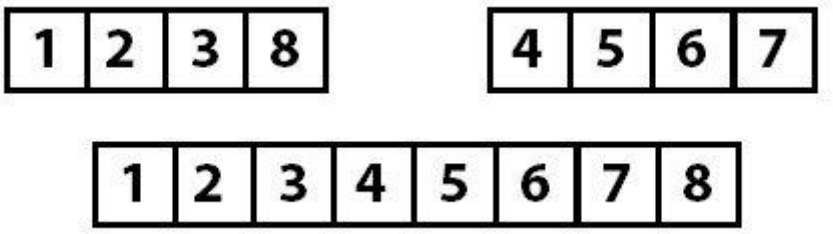

Рисунок 1.5 - Зливаємо дві останні групи в єдиний відсортований масив.

Для роботи алгоритму ми повинні реалізувати 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
```

Візьмемо один з прикладів тренажеру, який реалізований за операціями.

Приклад: A={5,8,4,12}

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

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]=5

3.5.4 for i=2 to 1 false

3.5.6 for j=1 to 1

3.5.7 R[1]=A[2]=8

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], 5<=8 true

3.5.14 A[1] = L[1]=5

3.5.15 i=i+1=2

3.5.12 for k = 2 to 2

3.5.13 if L[2]<= R[1], ∞<=8 false

3.5.16 else A[2]=R[1]=8

3.5.17 j=j+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[1]=4

4.5.4 for i=2 to 1 false

4.5.6 for j=1 to 1

4.5.7 L[1]= A[1]=12

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] 4<=12 true

4.5.14 A[3]=L[1]=4

4.5.15 i=i+1=2

4.5.12 for k = 4 to 4

4.5.13 if L[2]<= R[2], ∞<=12 false

4.5.16 else A[4]=R[2]=12

4.5.17 j=1+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]=5

5.4 for i=2 to 2 true

5.5 L[2]= A[2]=8

5.4 for i=3 to 2 false

5.6 for j=1 to 2

5.7 R[1]=A[3]=4

5.6 for j=2 to 2 true

5.7 R[2]=A[4]=12

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] 5<=4 false

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

5.17 j=1+1=2

5.12 for k=2 to 4

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

5.14 A[2]=L[1]=5

5.15 i=1+1=2

5.12 for k = 3 to 4

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

5.14 A[3]=L[2]=8

5.15 i=2+1=3

5.12 for k = 4 to 4

5.13 if L[3]<=R[2] ∞<=12 false

5.16 else A[4]=R[2]=12

5.17 j=2+1=3

5.12 for k = 5 to 4 false

Завершення сортування.

3.2 Розробка блок-схеми алгоритму

Крок 1. Запуск тренажеру.

Крок 2. Користувач отримує випадковий приклад.

Крок 3. Тренажер має певну кількість запитань, якщо вони закінчилися (так), з’являється повідомлення про закінчення, якщо питання не закінчилися(ні), переходимо на крок 4.

Крок 4. Задається питання: який крок алгоритму?

Крок 5. Студент вводить відповідь. Якщо відповідь правильна, переходимо на крок 6, якщо відповідь не правильна, з’являється повідомлення про помилку і поява підказки. Користувач заново вводить відповідь.

Крок 6. Задається питання: яка буде відповідь даного кроку?

Крок 7. Студент вводить відповідь. Якщо відповідь правильна, тренажер повертаються до кроку 2, якщо відповідь не правильна, з’являється повідомлення про помилку і поява підказки. Користувач заново вводить відповідь.

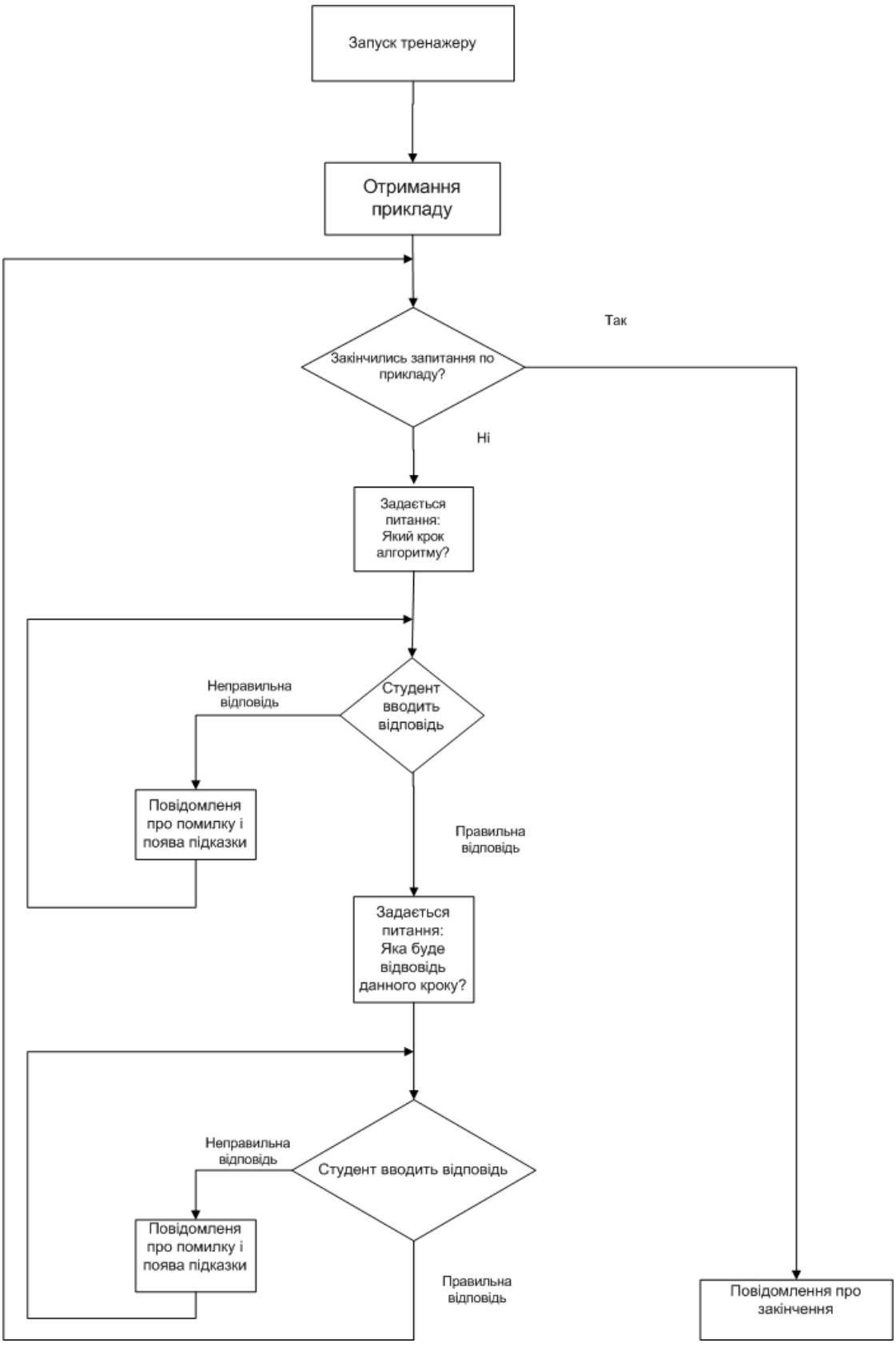

Рисунок 3.1 – Блок-схема алгоритму

3.3 Обґрунтування вибору програмних засобів для реалізації завдання роботи.

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

JavaFX дозволяє створювати інтерфейс з гарною графікою завдяки використанню апаратного прискорення графіки і можливостей GPU.

За допомогою JavaFX можна створювати програми для всіх операційних систем, а також для різних пристроїв таких, як смартфони, планшети, ТВ, тощо. Додаток на JavaFX буде працювати будь де, якщо встановлена виконуючого середовища Java (JRE).

JavaFX має великі можливості в порівнянні з іншими подібними платформами. Це великий набір елементів управління, можливості по роботі з мультимедіа, двомірної і тривимірною графікою, опису інтерфейсу за допомогою мови розмітки FXML, можливість стилізації інтерфейсу за допомогою CSS, та багато іншого.

На даний момент JavaFX є один з найкращих способів для створення графічних додатків за допомогою мови Java.

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 NetBeans.

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

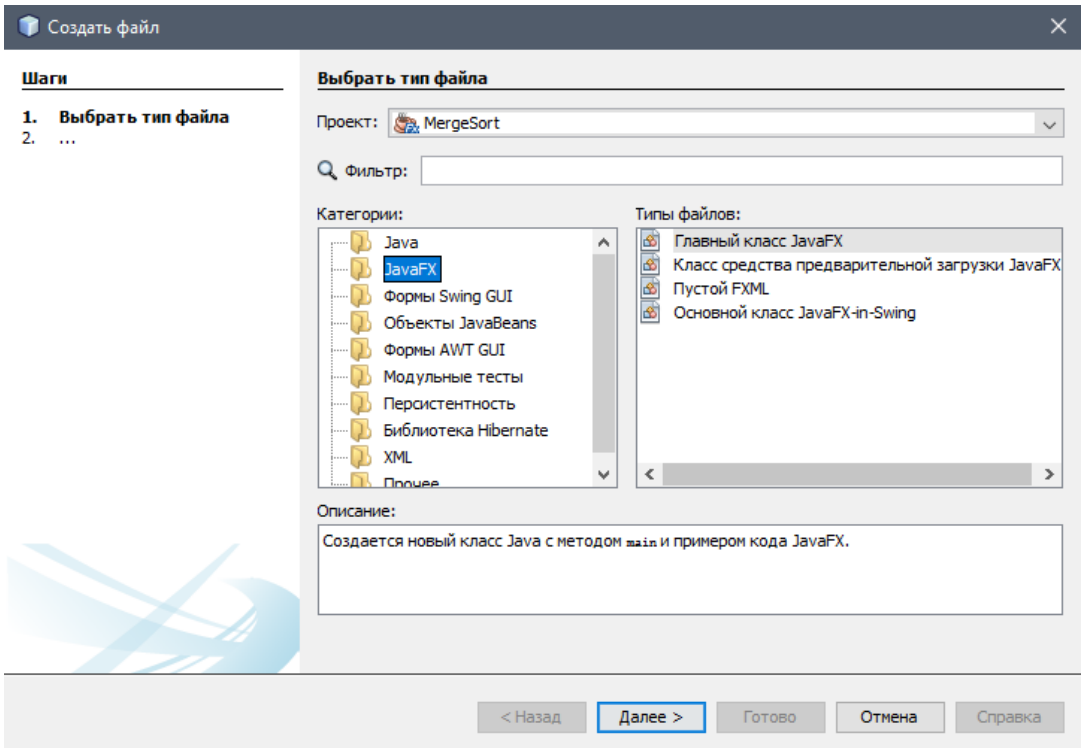

Рисунок 4.1 – Вибираємо тип файлу

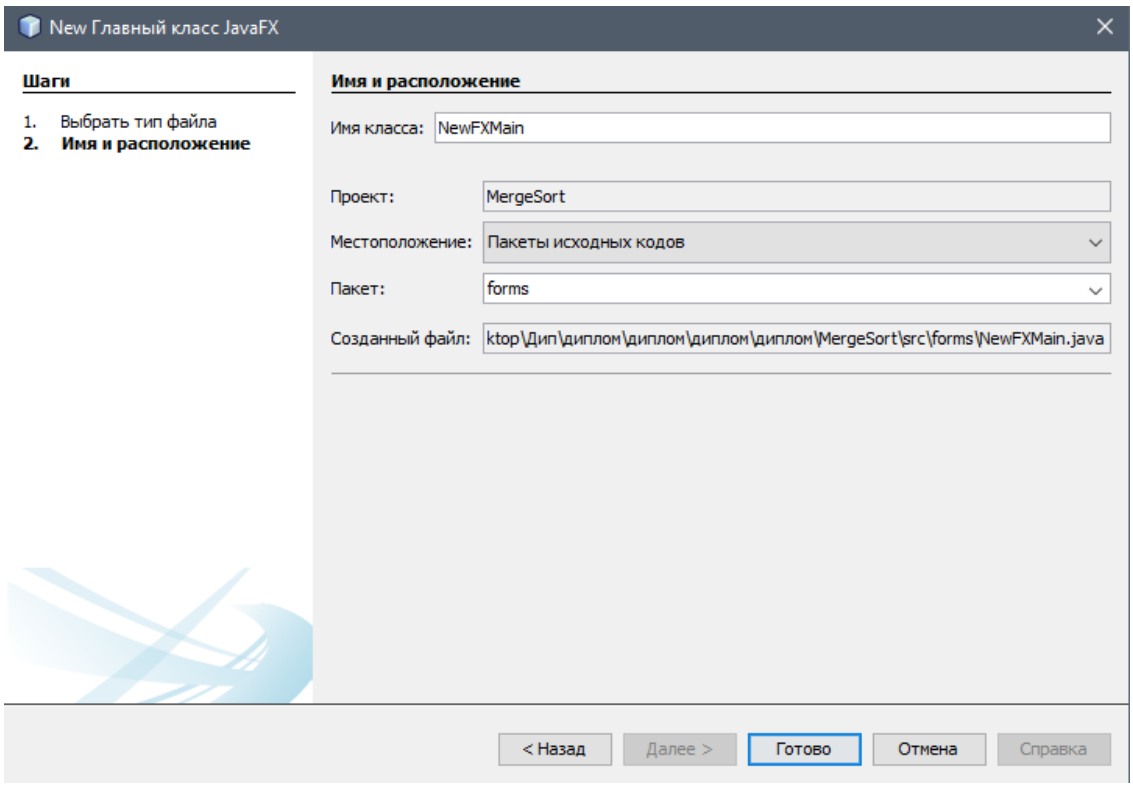

Рисунок 4.2 – Вибираємо ім’я та місце розташування.

Після створення файлу потрібно додати 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 дві форми Welcomeform.fxml та Mainform.fxml.

Приклад створення елементів у Welcomeform.fxml:

1. Створення тексту та його місце розташування на формі

```
<Label alignment="CENTER" prefHeight="88.0" prefWidth="400.0" text="Тренажер алгоритму сортування злиттям" textAlignment="CENTER" textOverrun="CLIP" wrapText="true">
        <font>
            <Font name="System Bold" size="18.0" />
        </font>
</Label>
```

2. Створення кнопки “Почати” та її місце розташування на формі

```
<HBox alignment="CENTER" prefHeight="60.0" prefWidth="400.0">
    <children>
      <Button fx:id="startButton" alignment="CENTER" mnemonicParsing="false" onMouseClicked="#goToMainForm" text="Почати" textAlignment="CENTER">
          <font>
            <Font size="18.0" />
          </font>
      </Button>
    </children>
</HBox>
```

Далі було створено два контролери WelcomeFormController.java та MainFormComtroller.java.

В WelcomeFormController.java було створено метод “goToMainForm”, який виконується після натискання на кнопку “Почати.

@FXML void goToMainForm (MouseEvent event) throws IOException.

В цьому методі ми отримуємо stage об’єкт.

Stage stage = (Stage) startButton.getScene().getWindow();

Після чого завантажуємо наш UI з файлу Mainform.fxml.

```
Parentroot = FXMLLoader.load(getClass().getResource("/forms/MainForm.fxml"));
        root.setId("pane");
```

Створюємо сцену та встановлюємо назву для вікна.

```
Scene scene = new Scene(root);
stage.setTitle("Merge Sort");
```

Далі додаємо сцену в stage.

```
stage.setScene(scene);
```

Та відображаємо stage і встановлюємо його в центрі екрану.

```
stage.show();
stage.centerOnScreen();
```

Також WelcomeFormController.java має клас “WelcomeFormController”, який має елемент інтерфейсу, кнопку “Почати”.

```
public class WelcomeFormController {
    @FXML
    private Button startButton;
}
```

В MainFormComtroller.java було створено одинадцять методів:

1. Метод ініціалізації.

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

2. Метод, який викликається після натискання кнопки “Відправити відповідь”.

```
void submit(MouseEvent event) {}.
```

3. Метод для очищення полей.

```
private void resetFields() {}.
```

4. Метод для встановлення цифр масива на екрані.

```
private void setNumbers() {}.
```

5. Метод для виводу полей для відповіді.

```
private void setFieldsForAnswers() {}.
```

6. Метод для отримання індексу вибраного кроку алгоритма.

```
private int getSelectedStepIndex() {}.
```

7. Метод для отримання масивів с іменами параметрів для вибраного крока алгоритма.

private String[] getSelectedStepParams() {}.

8. Метод для перевірки чи пусті поля.

private boolean checkFields() {}.

9. Метод для перевірки правильності введенної відповіді.

private boolean verifyAnswers() {}.

10. Метод для парсингу файлу

private void readFile() {}.

11. Метод для закінчення роботи с тренажером.

private void end() {}.

Також MainFormComtroller.java було створено два класи.

1. public class MainFormController {}, який містить всі елементи інтерфейсу.

2. private class Cell extends Rectangle {}, який відображає квадрати для місивів.

Також тренажер 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() {}

4.2 Опис програми

Після запуску тренажеру, перед студентом з’являється початкове вікно, де він може побачити назву тренажеру, кнопку “Почати” та відомість про автора, рис. (4.3).

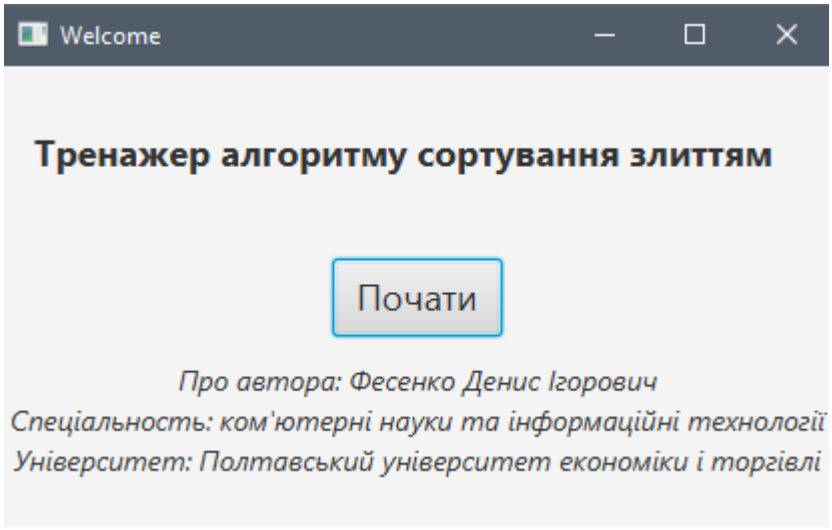

Рисунок 4.3 – Початкове вікно

Щоб почати трасування алгоритму, потрібно натиснути кнопку “Почати” після чого з’явиться вікно де розташований алгоритм, масив чисел, поля для відповідей та лог-файл рис. (4.4).

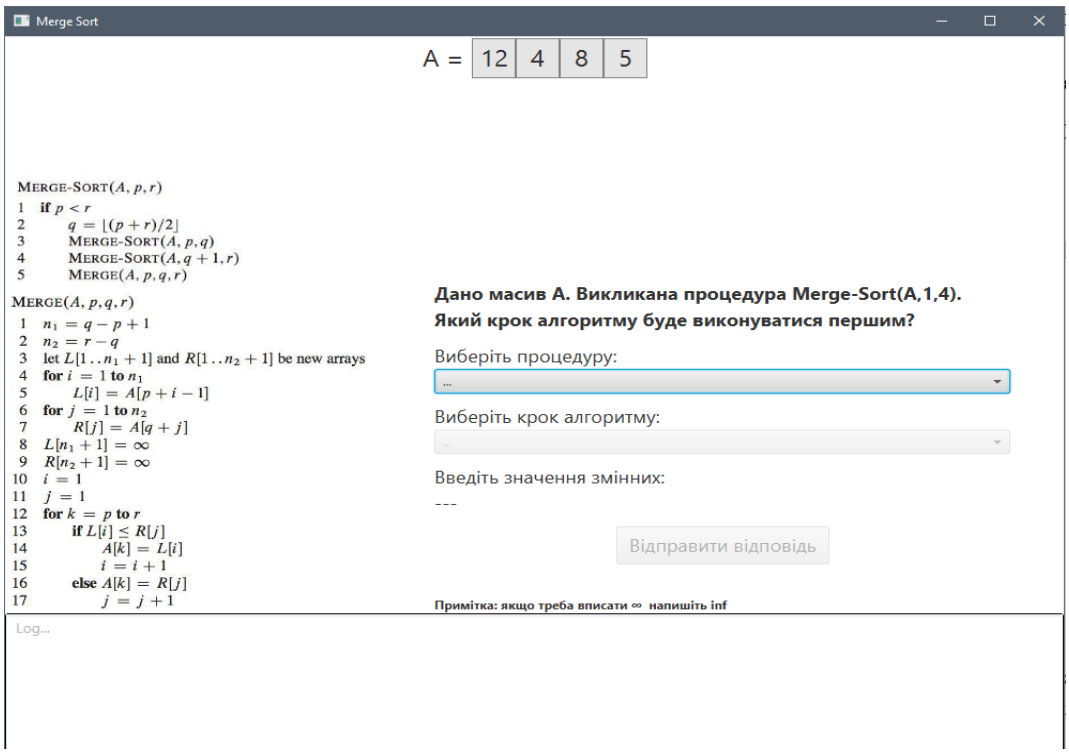

Рисунок 4.4 – Вікно з алгоритмом та прикладом

Далі студент повинен вибрати яка процедура та крок алгоритму буде виконуватися першим, після натиснення на поле “Виберіть процедуру” з’явиться список усіх можливих операцій, аналогічно с полем “Виберіть крок алгоритму”, рис. (4.5-4.7).

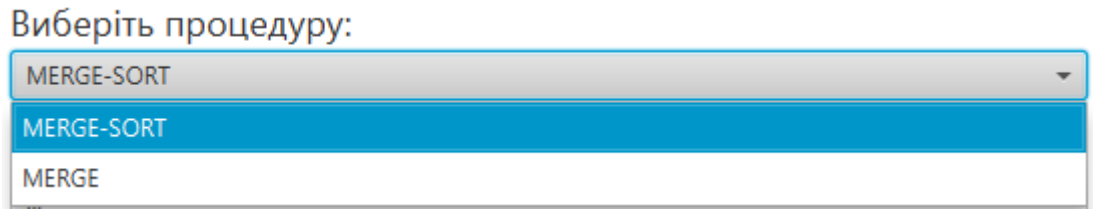

Рисунок 4.5 – Список поля “Виберіть процедуру”

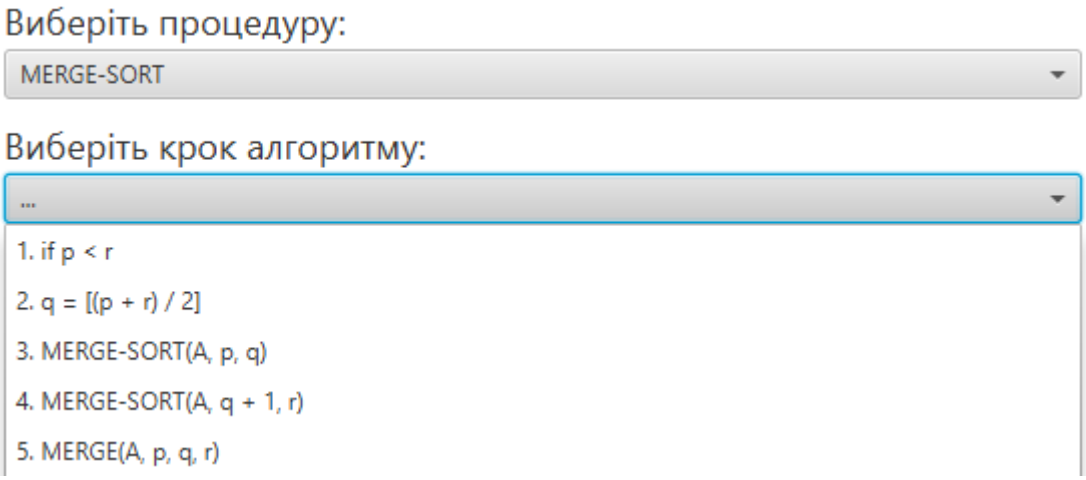

Рисунок 4.6 – Список поля “Виберіть крок алгоритму” процедури Merge-Sort

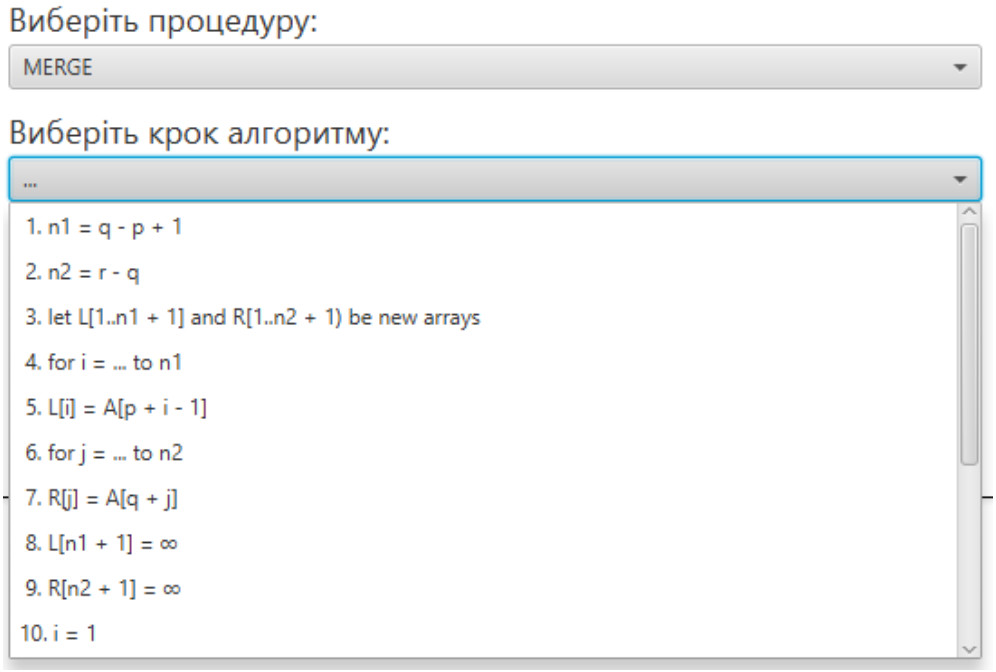

Рисунок 4.7 - Список поля “Виберіть крок алгоритму” процедури Merge

Після вибору процедури та кроку перед студентом з’явиться поля куди потрібно ввести відповідь та кнопка для відправки, рис. (4.8).

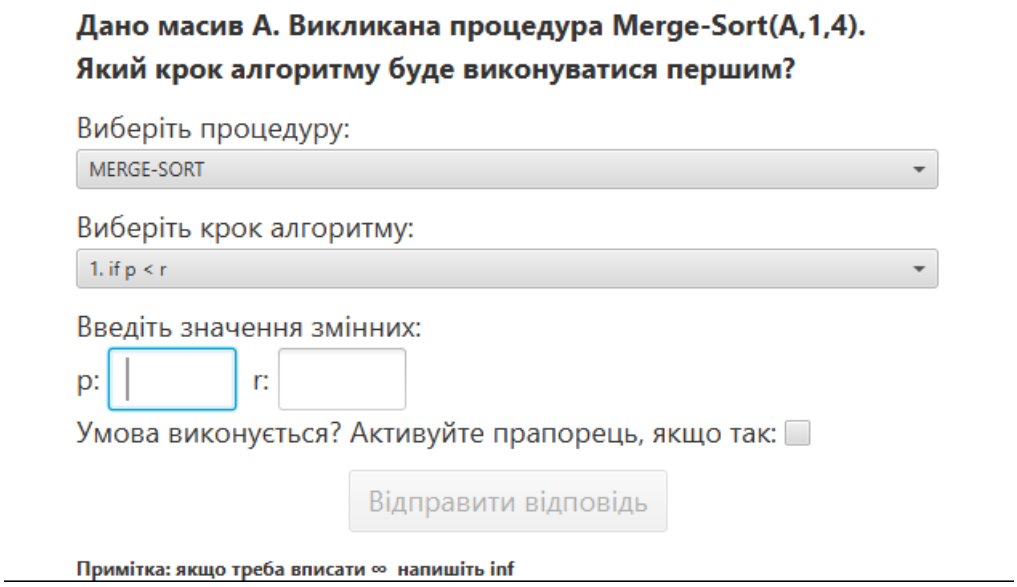

Рисунок 4.8 – Поля для відповіді та кнопка для відправки

Якщо користувач вводить відповідь неправильно, як на рис. (4.9), з’являється повідомлення про неправильну відповідь, якщо користувач вводить відповідь не правильно декілька раз з’явиться повідомлення з підказкою, як на рис. (4.10), також якщо користувач введе не цифрові значення або не виняткове слово, з’явиться повідомлення про введення не цифрових даних, рис. (4.11).

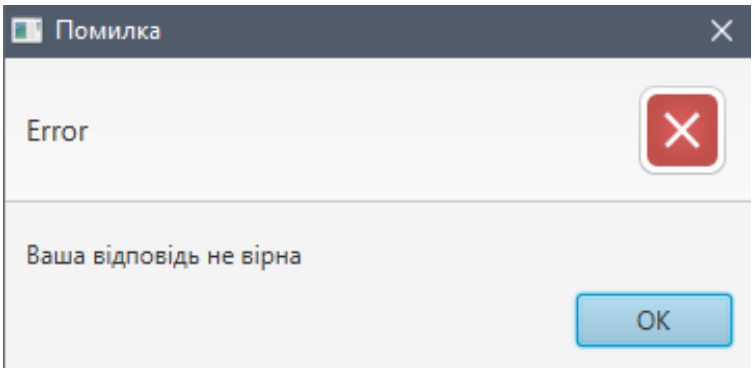

Рисунок 4.9 – Повідомлення про неправильну відповідь

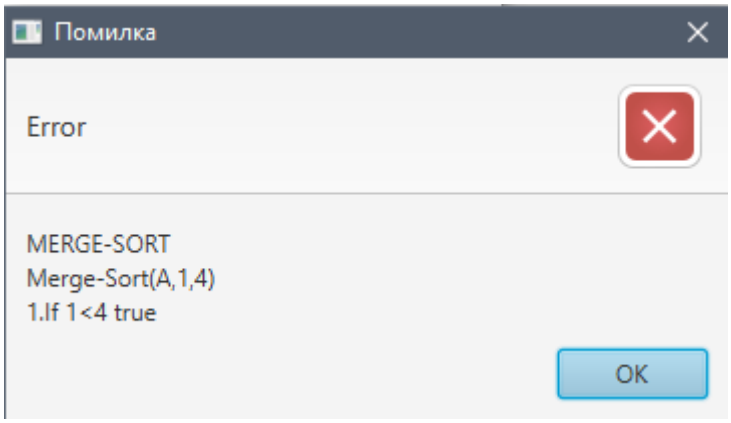

Рисунок 4.10 – Повідомлення з підказкою

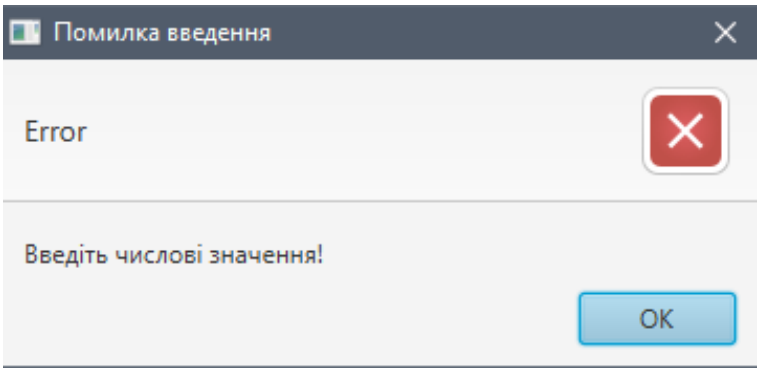

Рисунок 4.11 - повідомлення про введення не цифрових даних

Коли користувач вводить правильну відповідь, його відповідь зберігається в лог-файлі, як на рис. (4.12) та відповідати на наступні питання.

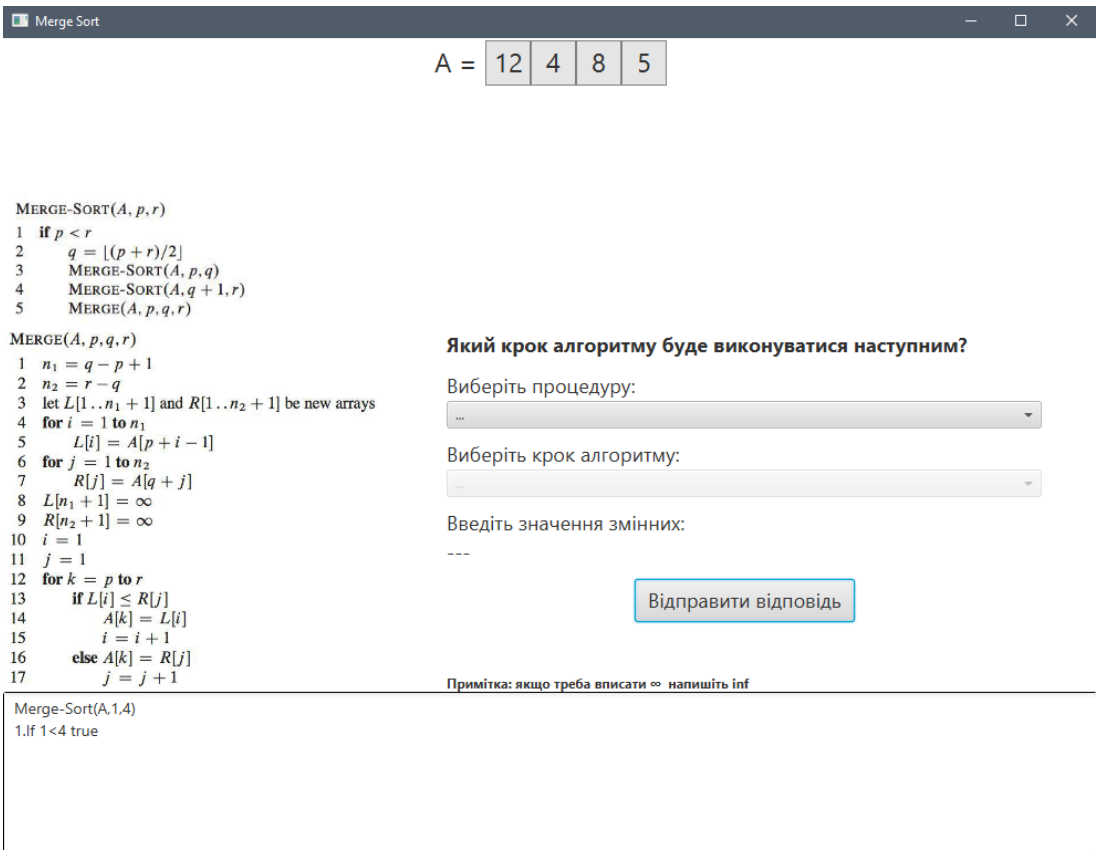

Рисунок 4.12 – Лог-файл

Після закінчення трасування алгоритму, студент отримає повідомлення про закінчення тренажеру, як на рис. (4.13), де буде розташована кнопка “ОК”, натиснув на яку, повідомлення закриється і всі поля заблокуються, але студент зможе переглянути увесь лог-файл та скопіювати його собі.

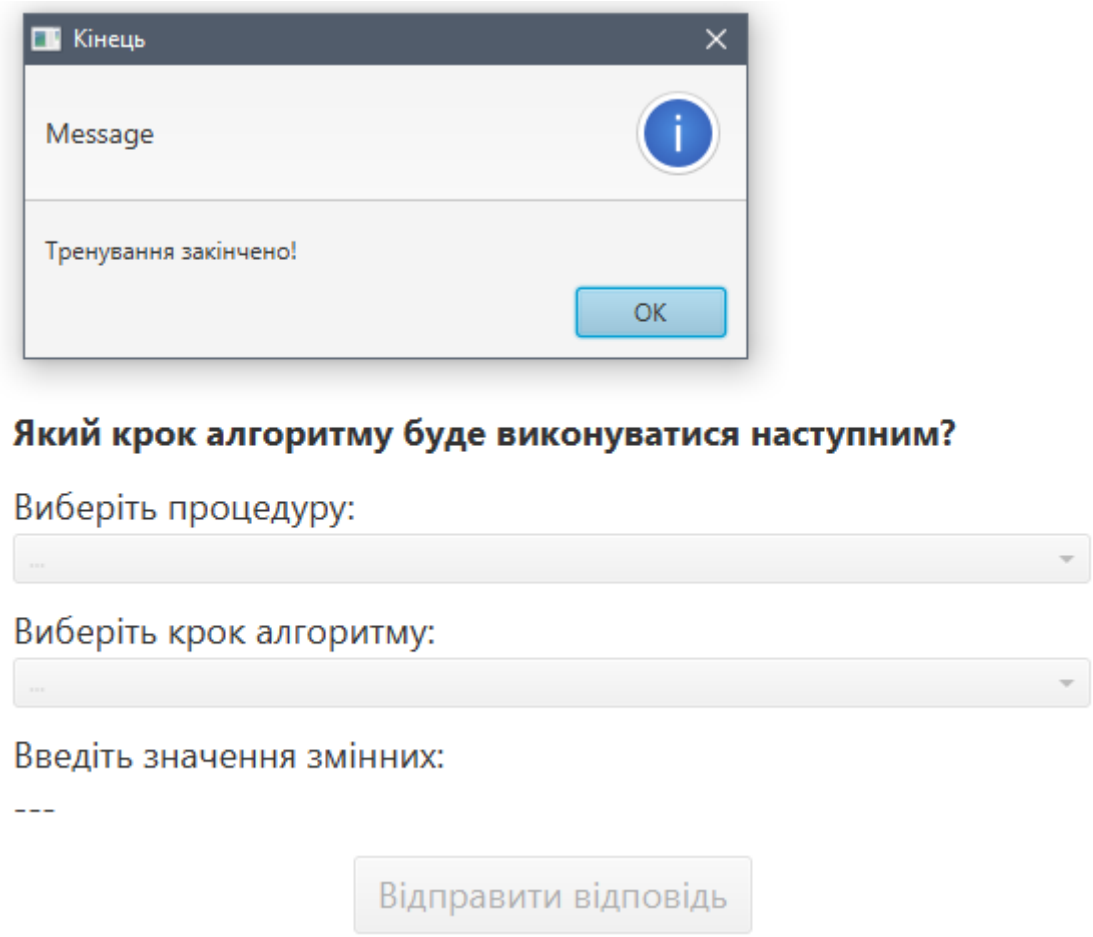

Рисунок 4.13 – Повідомлення про закінчення тренінгу

4.3 Перевірка валідності

Перевірка на валідність тренажера дає змогу зрозуміти чи виконує функції які було поставлено у меті та чи повністю він працездатний.

Для цього було проведено тестування тренажера.

1. Введення недопустимих символів, як на рис. (4.14).

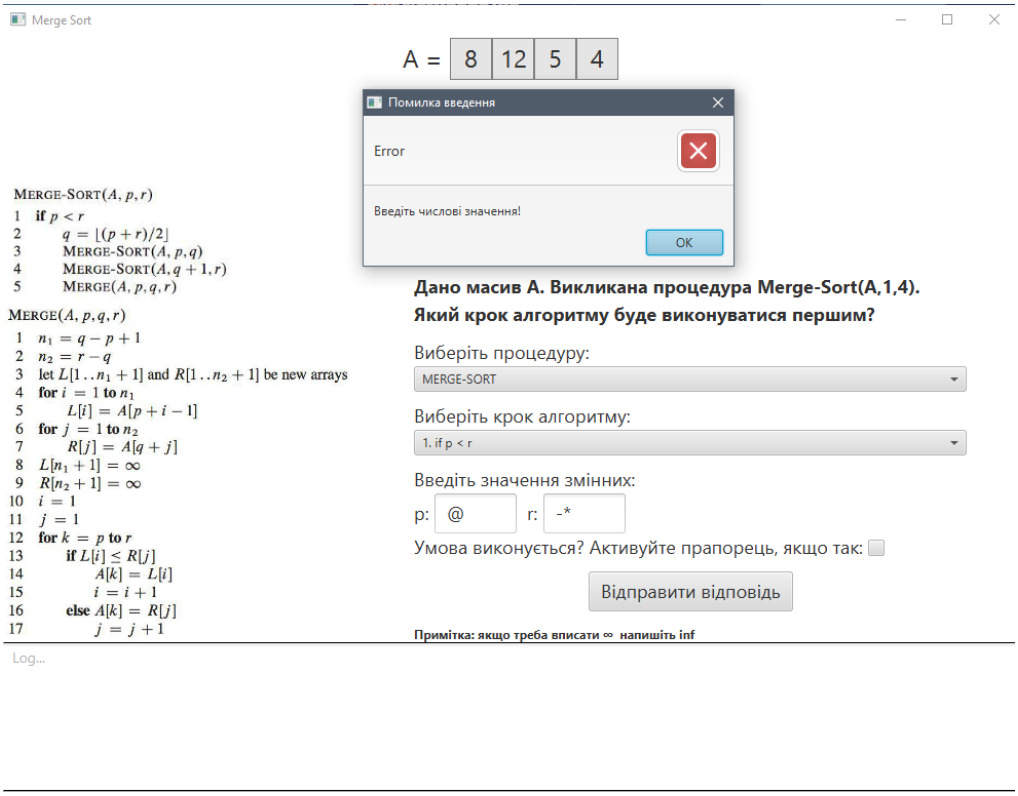

Рисунок 4.14 – Недопустимі символи

2. Вибір не правильної процедури або кроку алгоритму, як на рис. (4.15).

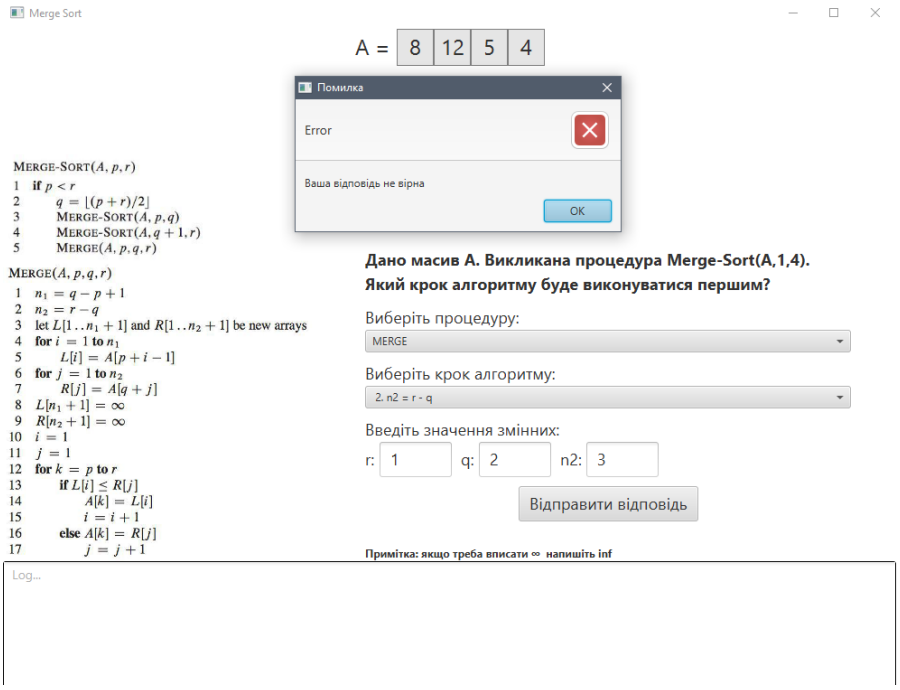

Рисунок 4.15 – Не правильний вибір процедури або кроку алгоритму

3. Вибір правильної процедури або кроку алгоритму та введення правильної відповіді, як на рис. (4.16-4.17).

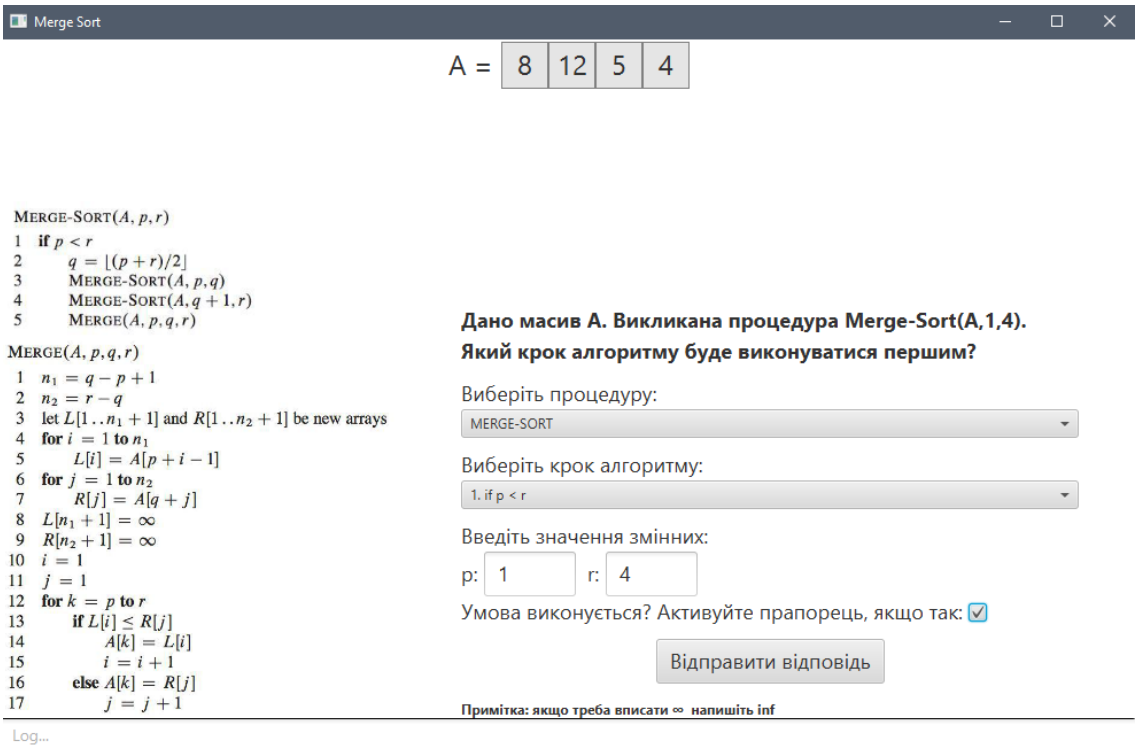

Рисунок 4.16 - Вибір правильної процедури або кроку алгоритму та введення правильної відповіді

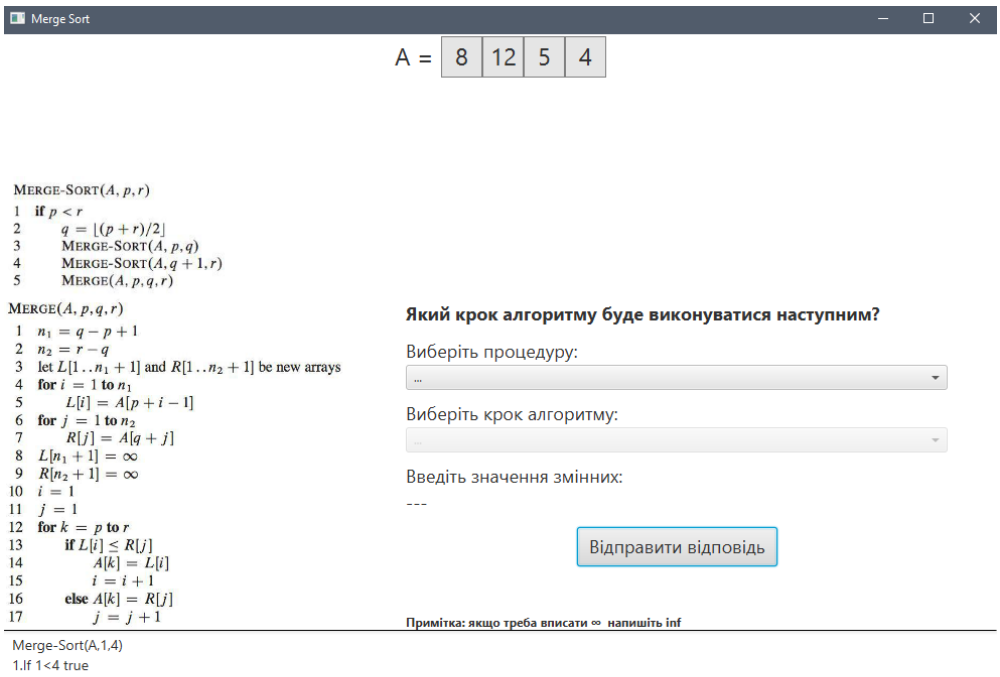

Рисунок 4.17 – Занесення правильної відповіді у лог-файл

Валідність даного тренажера означає , що він перевіряє тільки ті знання та навички, які повинен перевіряти. Тренажер містить тільки перевіренні приклади, як на рис. (4.18), що дає змогу не сумніватися в їх правильності та чітко перевіряє відповіді студента, щоб приймати тільки правильну відповідь.

```
"arrayA": [5, 8, 4, 12],
"tasks": [
    {
        "arrayA": [5, 8, 4, 12],
        "arrayL": [],
        "arrayR": [],
        "part": "MERGE-SORT",
        "step": 0,
        "answers": [1, 4, 1],
        "log": "Merge-Sort(A,1,4)\n1.If 1<4 true"
    },
    {
        "arrayA": [5, 8, 4, 12],
        "arrayL": [],
        "arrayR": [],
        "part": "MERGE-SORT",
        "step": 1,
        "answers": [1, 4, 2],
        "log": "2. Q=[(1+4/2)]=2"
    },
    {
        "arrayA": [5, 8, 4, 12],
        "arrayL": [],
        "arrayR": [],
        "part": "MERGE-SORT",
        "step": 2,
        "answers": [1, 2],
        "log": "3. Merge-Sort(A,1,2)"
    },
    {
        "arrayA": [5, 8, 4, 12],
        "arrayL": [],
        "arrayR": [],
        "part": "MERGE-SORT",
        "step": 0,
        "answers": [1, 2, 1],
        "log": "3.1. If 1<2 true"
    },
    {
        "arrayA": [5, 8, 4, 12],
        "arrayL": [],
        "arrayR": [],
        "part": "MERGE-SORT",
        "step": 1,
        "answers": [1, 2, 1],
        "log": "3.2. q = [(1+2/2)] = 1"
    },
    {
        "arrayA": [5, 8, 4, 12],
        "arrayL": [],
        "arrayR": [],
        "part": "MERGE-SORT",
        "step": 2,
        "answers": [1, 1],
        "log": "3.3 Merge-Sort(A,1,1)"
    },
```


Рисунок 4.18 – Перевірені приклади

4.4 Необхідна користувачу програмна інструкція

Для початку роботи з тренажером, необхідно встановити JDK 1.8. При запуску тренажера з’являється вікно по середині якого розташована кнопка “Почати”. Після натискання на кнопку з’являється вікно з основним інтерфейсом. Зверху розташований приклад, так званий масив чисел, зліва знаходиться алгоритм, за яким студент повинен зробити трасування алгоритму. Справа знаходяться два поля, в першому полі потрібно вибрати процедуру зі списку Merge-Sort чи Merge, у другому потрібно вибрати крок зі списку відповідно до процедури, яка була обрана. Далі з’являться поля, де потрібно вписати відповідь та натиснути кнопку “Відправити відповідь”. Якщо студент правильно на все відповів то його відповідь запишеться вниз, так званий лог-файл, де надалі будуть вноситись усі правильні відповіді поки трасування алгоритму не закінчиться. Після закінчення з’являться повідомлення про завершення тренінгу та з кнопкою “ОК”, після натиснення якого студент зможе переглянути увесь лог-файл та скопіювати за потребою.

ВИСНОВКИ

При виконанні бакалаврської роботи на основі розробленого алгоритму створено тренажер для аналізу алгоритму сортування злиттям.

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

Виконані основні вимоги для розробки навчального тренажеру, а саме:

1. Складено алгоритм тренажеру аналізу алгоритму сортування злиттям.
2. Складено блок-схему відповідно до розробленого алгоритму.
3. Програмно реалізовано навчальний тренажер у програмному середовищі NetBeans та на мові Java.
4. Перевірено працездатність тренажеру.

Виконані основні вимоги до роботи та використання навчального тренажеру студентом:

1. Інтерфейс програми інтуїтивно зрозумілий студенту.
2. При кожному введені відповіді, реалізувати перевірку даних та проінформувати студента, якщо відповідь неправильна та надати змогу відповісти ще раз та надати підказку, якщо студент багато раз відповів не правильно.
3. Присутній лог-файл для перегляду усіх відповідей.

Дотримання усіх вимог щодо розробки та роботи навчального тренажеру забезпечує надійне функціонування та ефективність його у використані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. 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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

7. Кормен Т. Алгоритмы: построение и анализ, 2-е изд./ Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн — М.: Вильямс, 2005. — 1296 с.

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

9. Герберт Шилдт. Java: Полное руководство, 9-е издание/Ш.Герберт,2015

10. Сортування злиттям [Електронний ресурс]: Echo. – Режим доступу: https://echo.lviv.ua/dev/6908