ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

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

Допускається до захисту

Завідувач кафедри _____О.О. Ємець
(підпис)

«_____»_____2020 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО БАКАЛАВСЬКОЇ РОБОТИ

на тему

ТРЕНАЖЕР З ТЕМИ «ОБРІЗАННЯ ОСНОВ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ТЕОРІЇ ПРОГРАМУВАННЯ» ТА РОЗРОБКА ЙОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Цехмейстер Вадим Олександрович
_____ «___»_____2020р.
(підпис)
Науковий керівник к.ф.-м.н., доц. Черненко Оксана Олексіївна
_____ «___»_____2020р.
(підпис)

ПОЛТАВА 2020р.

ЗМІСТ

ВСТУП

В епоху інформаційної доступності, коли Інтернет-технології розвиваються у різних напрямках, тим самим поліпшують життя людей. Важливим елементом в освіті є дистанційне навчання, а саме дистанційний курс. Не дивно, що всесвітня мережа влилась і у сферу освіти, та стала невід'ємним елементом у навчанні школярів та студентів. Завдяки чому було створено таке направлення у сфері освіти як дистанційне навчання, що дозволяє навчати велику кількість фахівців та викладати великі обсяги інформації для студентів, що в свою чергу дозволяє розвивати навички самостійності, та уміння фільтрувати інформацію. В свою чергу ретельно підібраний матеріал, та його баланс теоретичної та практичної частини, значно впливає на якість та швидкість засвоєння інформації студентом.

Міра і спосіб використання комп'ютерних технологій при підготовці дистанційного курсу значно впливає на ефективність його застосування. Світовий досвід показує, що використання динамічних об'єктів для створення наочних моделей процесів, адаптивне моделювання студента в багатьох випадках значно підвищує навчальний ефект. Для організації дійсно ефективного начального процесу дистанційного навчання необхідна систематична робота з оболонкою студента, так і викладача майже кожного дня на протязі всього терміну навчання. Дистанційне навчання досить актуальне, але основна проблема полягає у тому , що вимоги до фахівців змінюються щороку, виходячи з цього виникає необхідність удосконалення навчального процесу дистанційного направлення.

Метою роботи є програмна реалізація тренажера з теми «Обрізання основ» дистанційного начального курсу «Теорія програмування» та закріплення знань. Об'єктом розробки в даній роботі є процес дистанційного навчання математичним дисциплінам.

Предметом розробки є програмний продукт, що реалізує тренажер з теми «Обрізання основ» на мові програмування Java.

Тренажер готовий до використання в дистанційному курсі «Теорія програмування».

Робота складається з чотирьох розділів. У першому розділі розглянуто постановку задачі тренажеру. У другому розділі описано інформаційний огляд. У третьому розділі представлено теоретичну частину першого прикладу, теоретичну частину другого прикладу, алгоритмізацію задачі за темою роботи, блок-схему, яка підлягає програмуванню. У четвертому представлено – опис обраної мови програмування, опис процесу програмної реалізації, необхідну користувачу програми інструкцію.

Обсяг пояснювальної записки: 47 стор., в т.ч. основна частина - 32 стор., джерела - 11 назв.

1. ПОСТАНОВКА ЗАДАЧІ

1.1. Постановка основних вимог до змісту

Зміст роботи повинен відповідати завданню затвердженим керівником та завідувачем кафедри.

Розглянемо основні положення викладення матеріалу:

- чіткість та логічна послідовність викладення матеріалу;
- переконливість аргументації;
- стислість і точність формулювань, які виключають можливість неоднозначного тлумачення; коректність викладення результатів дослідження;
- обґрунтованість рекомендацій та пропозицій.

У роботі повинні бути відображеними:

- актуальність тематики та відповідність до сучасного стану науки, техніки і питань виробництва;
- обґрунтування вибраного напрямку досліджень, методів розв’язку задачі та їх порівнянь оцінки;
- аналіз та узагальнення існуючих результатів;
- розробка загальної методики проведення досліджень;
- характер і зміст виконаних теоретичних досліджень та розрахунків, методи досліджень;
- обґрунтування необхідності проведення експериментальних досліджень, принцип дії розроблених програм, характеристики цих програм, оцінка похибок розрахунків, отримані експериментальні дані; теоретичний (практичний) аналіз розроблених алгоритмів (програм);
- блок-схема програми (алгоритму);
- контрольні приклади;
- оцінка повноти розв’язку поставленої задачі;

- наукова або практична цінність виконаної роботи, виклад наукової новизни, якщо вона є.

1.2. Постановка задачі для реалізації програми-тренажера з теми «Обрізання основ»

Основним завданням роботи є розробка тренажеру з теми «обрізання основ» дистанційного курсу «Теорія програмування». Планується, що тренажер, який розробляється в рамках роботи використовуватиметься, як складова дистанційного курсу «Теорія програмування», а отже, необхідно врахувати можливість інтеграцій тренажеру до системи дистанційного навчання Moodle, на якій власне розміщуються дистанційні курси.

Розглянемо основні завдання роботи:

- провести вибір та обгрунтування мови програмування;
- розробити логічну схему інтерфейсу тренажеру з теми «Обрізання основ»;
- розглянути теоретичні відомості та специфіку застосування обрізання основ;
- розробити алгоритм тренажера з теми «Обрізання основ»;
- розробити навчальний тренажер з даної теми.

Тренажер, що розробляється повинен бути зручним у користуванні. Тому необхідно розробити можливість студента звернутися до теоретичного матеріалу з теми, що допоможе йому у проходженні завдання. Теоретичний матеріал має також містити приклади застосування обрізання основ.

Оскільки, тренажер розробляється з метою тренінгу умінь і навичок студента, то тренажер повинен передбачити виконанням студентами всіх кроків алгоритму розв’язування обрізання основ, а також автоматизацію перевірки правильності кожного з кроків та сповіщення про завершення тренінгу.

Розроблений тренажер повинен мати зрозумілий для користувача інтерфейс. Щоб досягти цього слід використовувати наступну структуру:

- стартова сторінка:
  - інформація про тренажер (тема, розробник, керівник);
  - перехід до теоретичного матеріалу;
  - перехід до тренінгу;
- задача для ознайомлення з роботою обрізання основ:
  - умова задачі;
  - поточна конфігурація роботи;
  - завдання (питання);
- результат проходження тренінгу:
  - кінцевий результат;
  - перехід до наступної задачі;
  - можливість закрити тренажер;
- задача для закріплення знань:
  - умова задачі;
  - завдання;
- результат проходження тренажера:
  - кінцевий результат;
  - можливість повторного виконання тренінгу;
  - можливість закрити тренажер.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Висхідний синтаксичний аналіз

У цьому розділі буде розглянутий основний метод висхідного синтаксичного аналізу, відомий як синтаксичний аналіз типу "перенесення/згортка" (shift-reduce) і називаний далі скорочено ПЗ-аналізом.

ПЗ-аналіз намагається будувати дерево розбору для вхідного рядка, починаючи з листів (знизу) і просуваючись в напрямку до кореня дерева (нагору). Цей процес можна розглядати як згортку рядка w до стартового символу граматики. На кожному кроці згортки (reduction step) деякий підрядок, що відповідає правої частини продукції, заміняється символом з лівої частини цієї продукції, і якщо на кожному кроці підрядки вибираються коректно, те ми одержуємо звернене праве породження. [2-3]

2.2. Основи

Неформально говорячи, основа або дескриптор (handle) рядка – це підрядок, що збігається із правою частиною продукції й згортка якої в ліву частину продукції являє собою один крок зверненого правого породження. У багатьох випадках крайній зліва підрядок β, що відповідає правої частини деякої продукції A→β не є основою, оскільки згортка відповідно до продукції A → β приводить до рядка, що не може бути згорнутий до стартового символу. Якщо в прикладі 1 ми замінимо в другому рядку aAbcde символ b нетерміналом A, то одержимо рядок aAAcde, що не може бути згорнутий в S. Із цієї причини нам варто дати більше точне визначення основи.

Якщо говорити формально, основа правосентенціальної форми γ є продукцією A→β і позицією рядка β в γ, такими, що β може бути замінена нетерміналом A для одержання попередньої правосентенціальної форми в

правому породженні γ. Таким чином, якщо $S \underset{rm}{\rightarrow}^{*} \alpha A w \underset{rm}{\rightarrow} \alpha\beta w$, то A→β у позиції після a являє собою основу рядка αβw. Рядок w праворуч від основи містить тільки термінальні символи. Помітимо, що граматика може бути неоднозначної, з декількома правими породженнями αβw. Якщо граматика однозначна, то кожна правосентенціальна форма граматики має рівно одну основу.

У наведеному вище прикладі abbcde являє собою правосентенциальную форму, основою якої є A→b у позиції 2. Аналогічно aAbcde являє собою правосентенциальную форму, дескриптор якої – A→Abc у позиції 2. Іноді ми будемо говорити "підрядок β являє собою основу αβw", якщо позиція β і продукція A →β визначаються однозначно.

На рис. 2.1 зображена основа A→β у дереві розбору правосентенціальної форми αβw. Основа представляє крайнє зліва завершене поддерево, що складається з вузла й всіх його нащадків. На рис.1 вузол A – нижній крайній зліва внутрішній вузол, всі нащадки якого перебувають у дереві. Згортку β до A в αβw можна представити як "обрізку основи", тобто видалення з дерева розбору всіх нащадків A. [4-6]

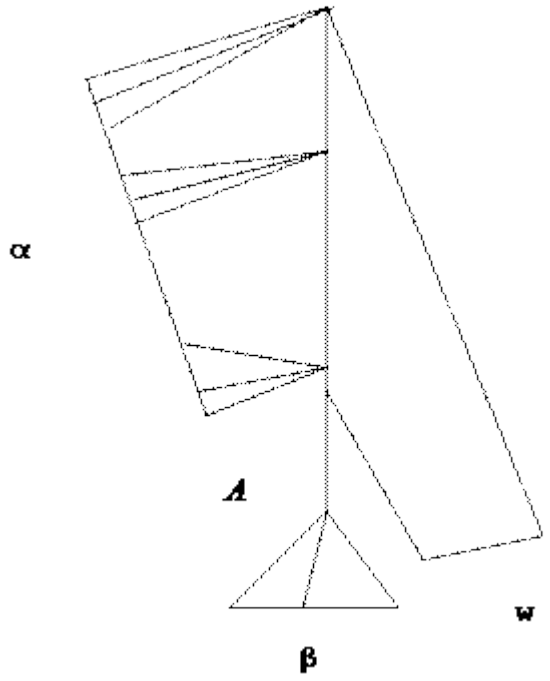

Рисунок 2.1. Основа A→β у дереві розбору αβw

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Теоретична частина першого прикладу

Ми скануємо рядок abbcde у пошуках підрядка, що відповідає правої частини якої-небудь продукції. Такими підрядками є b й d. Виберемо крайнє зліва b і замінимо його нетерміналом A, що являє собою ліву частину продукції A →b; таким чином, одержимо рядок aAbcde.

Тепер правим частинам продукцій відповідають підрядки Abe, b й d. Хоча b й є крайнім зліва підрядком, що відповідає правої частини однієї із продукцій, виберемо для заміни підрядка Abc і замінимо її нетерміналом A в відповідності із продукцією A→Abc. У результаті одержимо рядок aAde. Заміняючи d на B, ліву частину продукції B →d, одержуємо aABe, що відповідно до першої продукції заміняється стартовим символом S.

Отже, послідовність із чотирьох згорток дозволяє привести рядок abbcde до стартового символу S. Ці скорочення являють собою звернене (тобто записане у зворотному порядку) праве приведення S → aABe → aAde → aAbcde → abbcde. [3, 7-9]

3.2. Теоретична частина другого прикладу

Для зручності ми позначили підрядковими індексами id_1 і підкреслили основу кожної правосентенціальної форми. Наприклад, id_1 являє собою основу правосентенціальної форми $id_1+id_2*id_3$, оскільки id є правою частиною продукції E→id, і заміна id_1 на E приведе до попередньої правосентенціальної форми $E+id_2*id_3$. Звернемо увагу на те, що рядок справа від основи складається тільки з термінальних символів.

Звернене праве породження може бути отримане за допомогою "обрізки основ". Ми починаємо процес із рядка терміналів w, що хочемо проаналізувати. Якщо w – речення розглянутої граматики, то $w=\gamma_n$,

де γ_n – n-а правосентенціальна форма деякого, ще невідомого правого породження $S = \gamma_0 \underset{rm}{\rightarrow} \gamma_1 \underset{rm}{\rightarrow} \gamma_2 \underset{rm}{\rightarrow} \dots \underset{rm}{\rightarrow} \gamma_{n-1} \underset{rm}{\rightarrow} \gamma_n \underset{rm}{\rightarrow} w$.

Для відтворення цього породження у зворотному порядку ми знаходимо основу β_n, в γ_n і заміняємо її лівою частиною продукції $A_n \rightarrow \beta_n$ для одержання (n-1)-у сентенціальної форми γ_{n-1}. Помітимо, що поки ми не знаємо, яким чином шукати основи, але незабаром познайомимося з відповідними методами.

Потім ми повторюємо описаний процес, тобто знаходимо в γ_n основу β_{n-1} і згортаємо її для одержання правосентенціальної форми γ_{n-2}. Якщо після чергового кроку правосентенціальна форма містить тільки стартовий символ S, ми припиняємо процес і повідомляємо про успішне завершення аналізу. Звернена послідовність продукцій, використаних у згортках, являє собою праве породження вхідного рядка. [3, 7-9]

Для вхідного рядка id1+id2∗id3 послідовність згортань наведена в табл. 3.1.

Таблиця 3.1 – Згортання для рядка $id_1+id_2*id_3$

Правосентенціальна форма	Основа	Згортаюча продукція
$id_1+id_2*id_3$	id_1	E →id
$E+id_2*id_3$	id_2	E →d
$E+E*id_3$	id_3	E →id
E+E*E	E*E	E → * E
E+E	E+E	E →E+E
E		

3.3. Алгоритмізація задачі за темою роботи

Задача №1.

Крок 0. Виведення умови.

Розглянемо граматику:

(1) S →aABe;

(2) A →Abc;

(3) A →b;

(4) B → d.

Рядок abbcde зводиться до S за допомогою наступних кроків:

S → aABe→ aAbcde→ abbcde→ aAde;

Крок 1.Виходячи з наведеної умови задачі оберіть початкову точку:

А) A;

Б) B;

В) S.

Якщо відповідь вірна, виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: B»

Крок 2. Виходячи з початкової точки S, наступним кроком буде:

А) S→aABe;

Б) S→aBe;

В) S→aAe.

Якщо відповідь вірна, виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз –

«Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: А»;

Крок 3. Виходячи з умови задачі, та правила лівого породження, виберіть яку літеру потрібно замінити S→aABe:

А) А;

Б) В;

В) а.

Якщо відповідь вірна, виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: А»

Крок 4. Виходячи з умови задачі, та правила лівого породження, виберіть яку літеру потрібно замінити aABe→ aAbcBe:

А) aABe→aAbBe;

Б) aABe→aAbcBe;

В) aABe→aBeAb.

Якщо відповідь вірна, виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: Б»

Крок 5. Виходячи з умови задачі, на яку літеру потрібно замінити у виразі «aAbcBe»:

А) А;

Б) В;

В) а.

Якщо відповідь вірна то виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: А»

Крок 6. Виходячи з умови задачі, та правила лівого породження, виберіть яку літеру потрібно замінити aABe→ aAbcBe:

А) aAbcBe літеру А на b, що матиме вигляд abbcBe;

Б) aAbcBe літеру В на a, що матиме вигляд abbcae;

В) aAbcBe літеру А на a, що матиме вигляд aabcBe.

Якщо відповідь вірна то виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: А»

Крок 7. Виходячи з умови задачі, та правила лівого породження, виберіть яку літеру потрібно замінити aABe→ aAbcBe→ abbcBe:

А) abbcBe літеру a на b, що матиме вигляд bbbcBe;

Б) abbcBe літеру В на d, що матиме вигляд abbcde;

В) abbcBe літеру c на a, що матиме вигляд abbaBe.

Якщо відповідь вірна то виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: Б».

По завершенню тренажера виводиться шкала оцінювання. Після чого користувач може завершити тестування та перейти до теорії, або вийти.

Також може розпочати тестування знову, або приступити до іншого завдання.

Задача №2

Крок 0. Виведення умови .

Розглянемо наступну граматику:

(1) E →E+E

(2) E→ E*E

(3) E → (E)

(4) E → id

і праве породження рядка id1+id2∗id3 запишеться так:

$E \underset{rm}{\rightarrow} E+E \underset{rm}{\rightarrow} E+E*E \underset{rm}{\rightarrow} E+E*id_3 \underset{rm}{\rightarrow} E+id_2*id_3 \underset{rm}{\rightarrow} id_1+ id_2*id_3$.

Крок 1. Виходячи з наведеної умови задачі оберіть початкову точку:

А) E;

Б) A;

В) B.

Якщо відповідь вірна, виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: A»

Крок 2. Виходячи з правосенціальної форми E+E, вихід в згортаючу продукцію буде:

А E → (E);

Б) E → E+E;

В) $E \rightarrow E+E*id_2$.

Якщо відповідь вірна, виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: Б».

Крок 3. Виходячи з умови задачі, згортаючої продукції виберіть наступний варіант для дії правосентенціальної форми: E →E+E :

А) E→ E+E*E;

Б) E→ E+E – E;

В) E→ E+E*id.

Якщо відповідь вірна, виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: А».

Крок 4. Виходячи з умови задачі, виберіть правосентенціальну форму для дії згортаючої продукції E+E*E, основа якого являє E*E:

А) $E+id_2*id_3$;

Б) $E+id_3$-E;

В) E→E*E.

Якщо відповідь вірна, виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: В».

Крок 5. Виходячи з умови задачі, виберіть згортаючу продукцію з правосентенціальної форми $E+E*id_3$:

А) $E+id_2*id_3$;

Б) E+id_3-E;

В) E → id.

Якщо відповідь вірна, виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: В».

Крок 6. Виходячи з умови задачі, виберіть основу з згортаючої продукції E → id:

А) id_3;

Б) id;

В) id_2.

Якщо відповідь вірна, виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: А».

Крок 7. Виходячи з умови задачі, виберіть згортаючу продукцію з правосентенціальної форми E+id_2*id_3:

А) E → d;

Б) E;

В) E →E.

Якщо відповідь вірна, виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: А».

Крок 8. Виходячи з умови задачі, виберіть основу з правосентенціальної форми $id_1+id_2*id_3$:

А) id_1;

Б) id_2;

В) id_3.

Якщо відповідь вірна, виконується перехід на наступний крок. Якщо ні, то виводиться повідомлення про помилку: допущено перший раз – «Помилка! Спробуйте знову.», в іншому випадку – «Помилка! Правильна відповідь: А».

По завершенню тренажера виводиться шкала оцінювання. Після чого користувач може завершити тестування та перейти до теорії, або вийти. Також може розпочати тестування знову, або приступити до іншого завдання.

3.4. Розробка блок-схеми, яка підлягає програмуванню

На рисунку 3.1 зображено блок-схему алгоритму роботи тренажера.

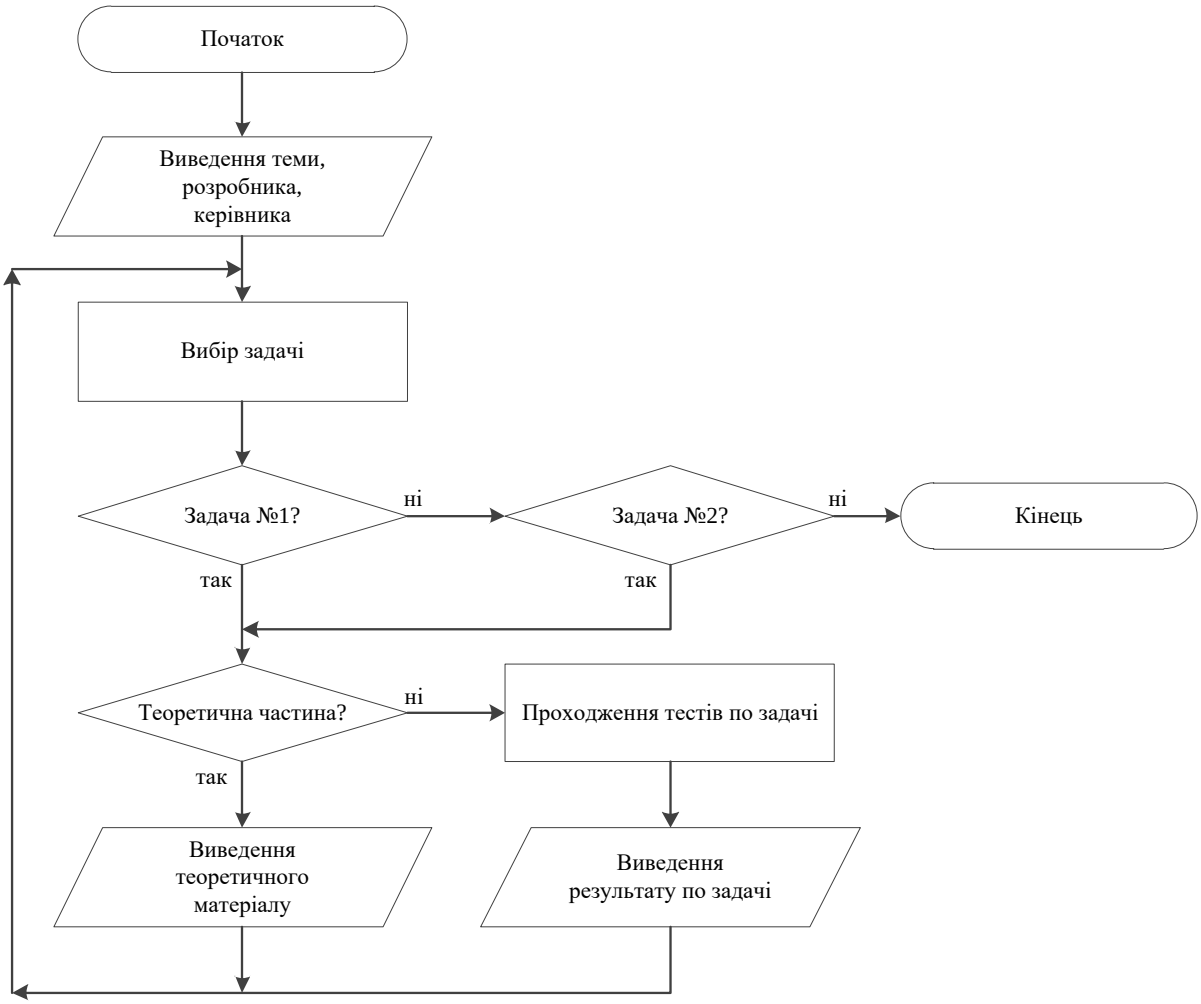

Рисунок 3.1 – Блок-схема алгоритму роботи тренажера

4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис обраної мови програмування

Java — об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java. Зараз мовою займається компанія Oracle, яка придбала Sun Microsystems у 2009 році. Синтаксис мови багато в чому схожий на C та C++. У офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Oracle надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцензією GNU General Public License.

Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням. За необхідності таких дій java дозволяє викликати підпрограми, написані іншими мовами програмування.

Під «незалежністю від архітектури» мається на увазі те, що програма, написана на мові Java, працюватиме на будь-якій підтримуваній апаратній чи системній платформі без змін у початковому коді та перекомпіляції.

Цього можна досягти, компілюючи початковий Java код у байт-код, який є спрощеними машинними командами. Потім програму можна виконати на будь-якій платформі, що має встановлену віртуальну машину Java, яка інтерпретує байткод у код, пристосований до специфіки конкретної

операційної системи і процесора. Зараз віртуальні машини Java існують для більшості процесорів і операційних систем.

Стандартні бібліотеки забезпечують загальний спосіб доступу до таких платформозалежних особливостей, як обробка графіки, багатопотоковість та роботу з мережами. У деяких версіях задля збільшення продуктивності JVM байт-код можна компілювати у машинний код до або під час виконання програми.

Основна перевага використання байт-коду — це портативність. Тим не менш, додаткові витрати на інтерпретацію означають, що інтерпретовані програми будуть майже завжди працювати повільніше, ніж скомпільовані у машинний код, і саме тому Java одержала репутацію «повільної» мови. Проте, цей розрив суттєво скоротився після введення декількох методів оптимізації у сучасних реалізаціях JVM.

Одним із таких методів є just-in-time компіляція (JIT, що перетворює байт-код Java у машинний під час першого запуску програми, а потім кешує його. У результаті така програма запускається і виконується швидше, ніж простий інтерпретований код, але ціною додаткових витрат на компіляцію під час виконання. Складніші віртуальні машини також використовують динамічну рекомпіляцію, яка полягає в тому, що віртуальна машина аналізує поведінку запущеної програми й вибірково рекомпілює та оптимізує певні її частини. З використанням динамічної рекомпіляції можна досягти більшого рівня оптимізації, ніж за статичної компіляції, оскільки динамічний компілятор може робити оптимізації на базі знань про довкілля періоду виконання та про завантажені класи. До того ж він може виявляти так звані гарячі точки (англ. hot spots) — частини програми, найчастіше внутрішні цикли, які займають найбільше часу при виконанні. JIT-компіляція та динамічна рекомпіляція збільшує швидкість Java-програм, не втрачаючи при цьому портативності.

Існує ще одна технологія оптимізації байткоду, широко відома як статична компіляція, або компіляція ahead-of-time (AOT). Цей метод

передбачає, як і традиційні компілятори, безпосередню компіляцію у машинний код. Це забезпечує хороші показники в порівнянні з інтерпретацією, але за рахунок втрати переносності: скомпільовану таким способом програму можна запустити тільки на одній, цільовій платформі.

Швидкість офіційної віртуальної машини Java значно покращилася з моменту випуску ранніх версій, до того ж, деякі випробування показали, що продуктивність JIT-компіляторів у порівнянні зі звичайними компіляторами у машинний код майже однакова. Проте ефективність компіляторів не завжди свідчить про швидкість виконання скомпільованого коду, тільки ретельне тестування може виявити справжню ефективність у даній системі.

Апле́т (англ. Applet) — коротка комп'ютерна програма, що функціонально розширює можливості основної програми, або інтернет-застосування. Наприклад, додає в веб-сторінку функцію анімації. Для створення аплетів в мережі Інтернет використовується мова програмування Java.

Java-аплет забезпечує можливість використовувати в World Wide Web «виконуваний вміст» шляхом виконання в гіпертекстовому документі міні-програм, відомих як аплети. Незвичайним є формат готового до виконання коду, у який транслюються аплети Java, — це апаратно-незалежний побайтовий формат. Завдяки використанню такого нейтрального формату відкомпільовані Java-програми можна переміщати по Internet, оскільки вони не залежать від операційної системи, у якій виконуються [10].

4.2. Опис процесу програмної реалізації

Для роботи деяких елементів було підключено дві бібліотеки:

- CardLayout;
- JOptionPane.

```
import java.awt.CardLayout;
import javax.swing.JOptionPane;
```

Щоб програма працювала покроково відповідно алгоритму створено змінну int step. Для підрахунку допущених помилок використовується int err_count, a int error – кількість помилок на поточному кроці.

```
int step = 1;
int err_count = 0, error =0;
```

Також було розроблено функцію void ShowStep(int ex, int i) для виведення питання відповідно задачі і кроку алгоритму.

```
private void ShowStep(int ex, int i){
   if(ex == 2) {
      CardLayout cl =(CardLayout) Example2.getLayout();
      cl.show(Example2, "ex2step"+i);
   } else {
      CardLayout cl =(CardLayout) Example1.getLayout();
      cl.show(Example1, "ex1step"+i);
   }
}
```

Відображення спочатку одного повідомлення про помилку при першій невірній відповіді, а потім іншого, реалізовано за допомогою void ShowError(int i, String text).

```
private void ShowError(int i, String text) {
   if(i <= 1) {
      JOptionPane.showMessageDialog(rootPane, "<html>Помилка! Спробуйте знову.", "ПОМИЛКА", JOptionPane.WARNING_MESSAGE);
   } else {
```

```
            JOptionPane.showMessageDialog(rootPane, "<html>Помилка! Правильна відповідь: "+text, "ПОМИЛКА", JOptionPane.INFORMATION_MESSAGE);
        }
    }
```

Після цього було розроблено події для роботи кнопок. Кнопка «Переглянути теоретичну частину» відкриває сторінку з теоретичним матеріалом вибраної задачі.

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        CardLayout cl =(CardLayout) Main.getLayout();
        if(jRadioButton2.isSelected()) {
            cl.show(Main, "theory2");
        } else {
            cl.show(Main, "theory1");
        }
    }
```

Аналогічним чином працює «Розпочати практичну частину», але вона виводить вже умову задачі і питання.

```
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        CardLayout cl =(CardLayout) Main.getLayout();
        if(jRadioButton2.isSelected()) {
            cl.show(Main, "example2");
            cl =(CardLayout) Example2.getLayout();
            cl.show(Example2, "ex2step1");
        } else {
            cl.show(Main, "example1");
```

```
            cl =(CardLayout) Example1.getLayout();
            cl.show(Example1, "ex1step1");
        }
    }
```

Кнопка «<<<Повернутися назад» повертає користувача до головної сторінки або результатів тренажера в залежності від поточного кроку.

```
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{
        CardLayout cl =(CardLayout) Main.getLayout();
        if(step == 1) {
            cl.show(Main, "home");
        } else {
            cl.show(Main, "result");
        }
    }
```

Перевірку відповіді і перехід до наступного кроку виконує «Перейти далі>>>». Для кожної задачі вона відрізняється (див. Додаток А).

Також на сторінці з результатами кнопка «Розпочати практичну частину» видаляє всі відповіді, встановлює step = 1, кількість помилок обнуляє, а лише потім відбувається перехід до задачі.

```
    private void jButton8ActionPerformed(java.awt.event.ActionEvent evt)
{
        buttonGroup2.clearSelection();
        step = 1;
        error = 0;
        err_count = 0;
        CardLayout cl =(CardLayout) Main.getLayout();
        if(jRadioButton49.isSelected()) {
```

```
            cl.show(Main, "example2");
            cl =(CardLayout) Example2.getLayout();
            cl.show(Example2, "ex2step1");
        } else {
            cl.show(Main, "example1");
            cl =(CardLayout) Example1.getLayout();
            cl.show(Example1, "ex1step1");
        }
    }
```

Кнопка «Закрити тренажер» закриває тренажер.

```
    private void jButton9ActionPerformed(java.awt.event.ActionEvent evt)
{
        System.exit(0);
    }
```

4.3. Необхідна користувачу програми інструкція

Після запуску тренажера відображається головна сторінка. На ній виводиться тема тренажера, виконавець роботи і науковий керівник, надається вибір між задачами (рис. 4.1).

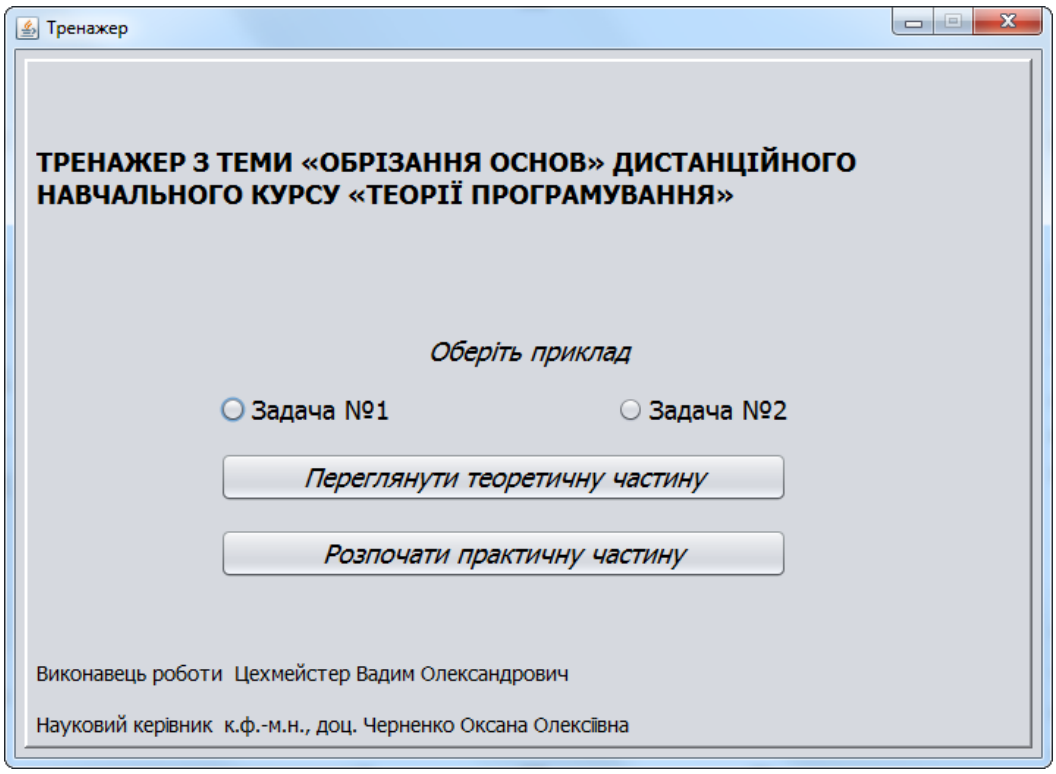

Рисунок 4.2 – Головна сторінка

Потрібно вибрати номер задачі та натиснути «Переглянути теоретичну частину» або «Розпочати практичну частину». Якщо було обрано перегляд матеріалу, то виведеться відповідна сторінка (рис. 4.3-4.4).

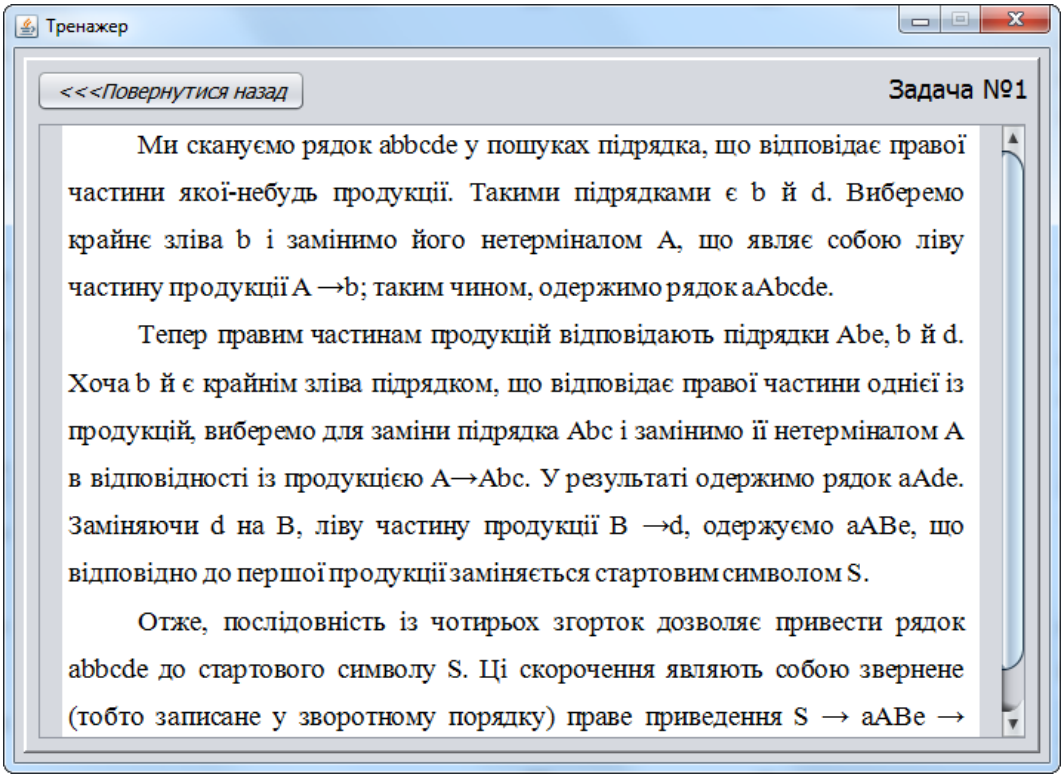

Рисунок 4.3 – Теоретична частина задачі №1

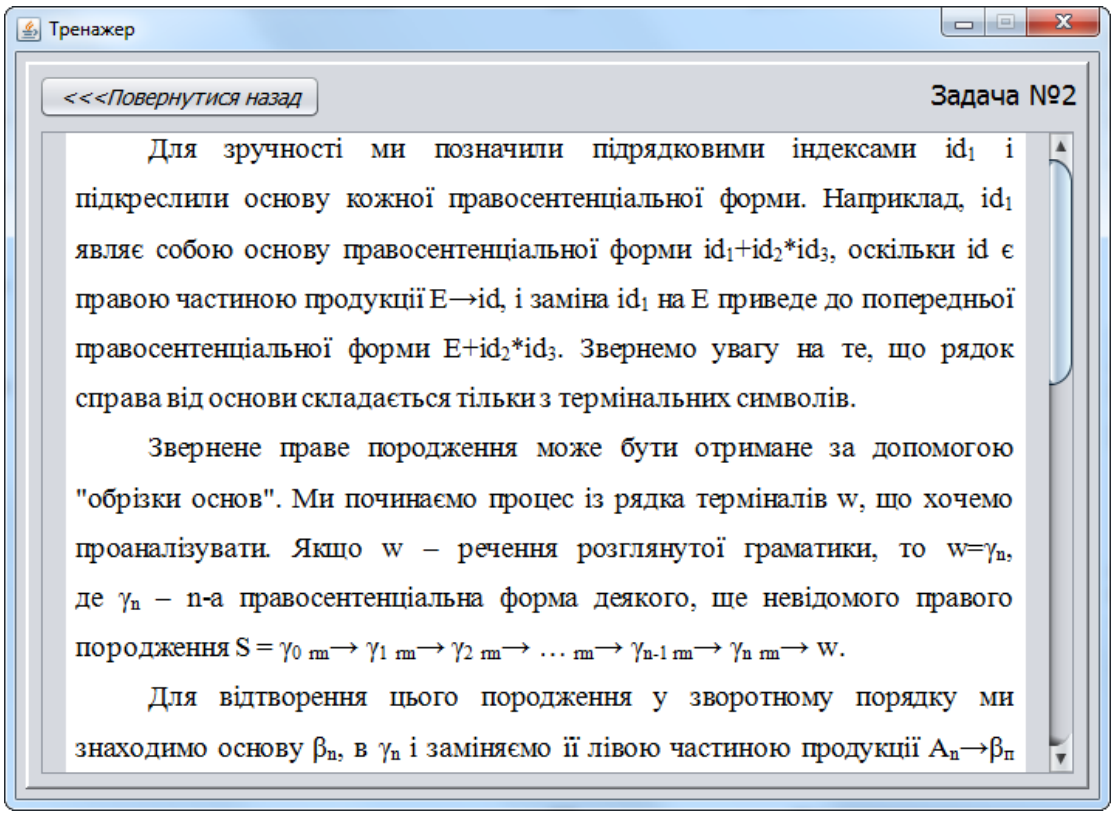

Рисунок 4.4 – Теоретична частина задачі №2

При виборі практичної частини відобразиться умова задачі і перше завдання. Необхідно вибрати один із варіантів (рис. 4.5).

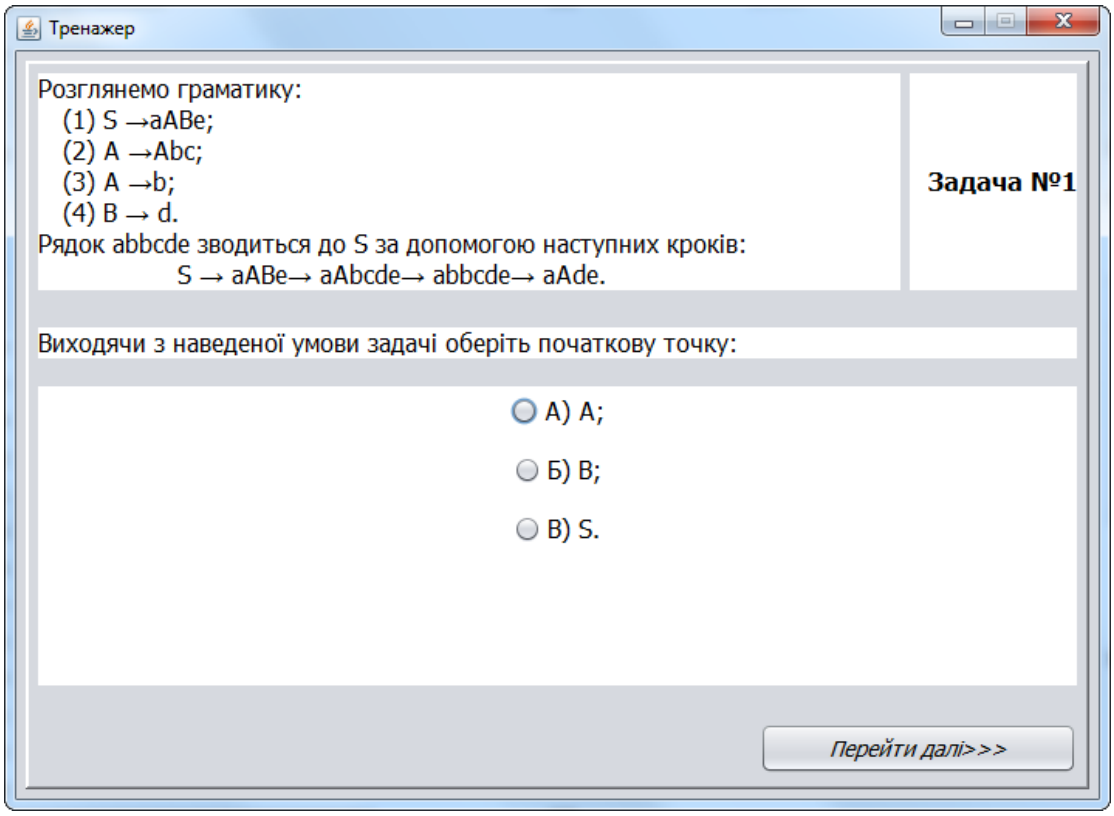

Рисунок 4.5 – Практична частина задачі №1

Якщо невірно відповісти перший раз, то з’явиться повідомлення з попередженням (рис. 4.6). В інший раз вже буде повідомлення про помилку (рис. 4.7).

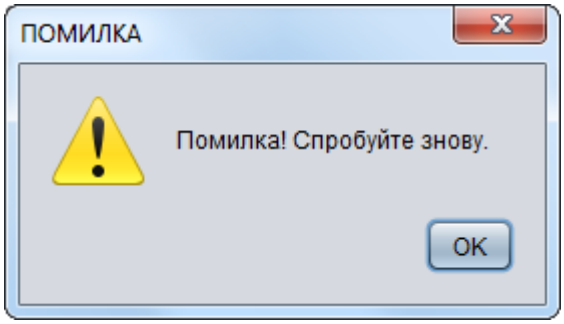

Рисунок 4.6 – Повідомлення з попередженням

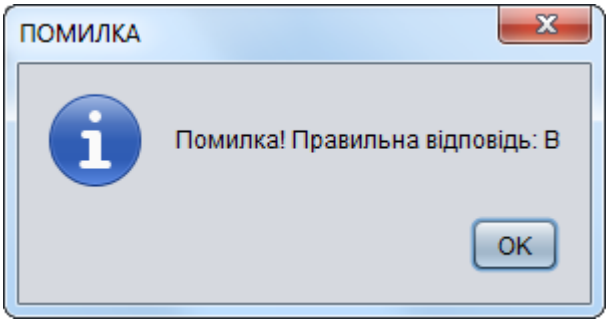

Рисунок 4.7 – Повідомлення про помилку

На останньому кроці виводиться повідомлення про завершення, кількість допущених помилок, пропонується знову вибрати задачу (рис. 4.8).

Друга задача розв’язується аналогічно розглянутій. Щоб закрити тренажер потрібно натиснути кнопку «Закрити тренажер».

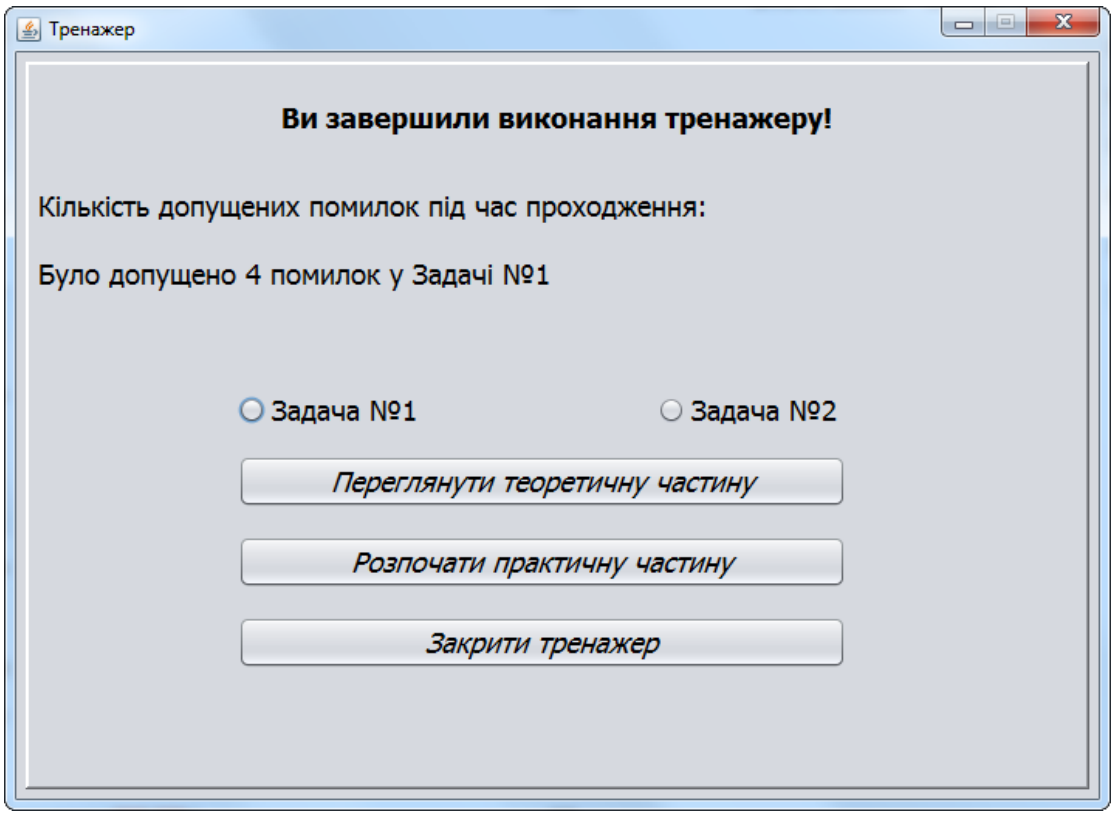

Рисунок 4.8 – Результати проходження

ВИСНОВКИ

Впровадження дистанційного навчання, та сучасних технологій в навчальний процес дозволяє підвищити ефективність навчання та якість формування вмінь студентів за допомогою віртуальних тренажерів.

Основною ціллю написання курсового і дипломного проекту є формування навичок самостійної творчої роботи і закріплення теоретичних знань, отриманих при вивченні дисципліни «Теорії програмування», а також суміжних загальнотеоретичних і спеціальних дисциплін, розвиток технічного мислення.

Планується, що тренажер, який розробляється в рамках роботи використовуватиметься, як складова дистанційного курсу «Теорія програмування», а отже, було врахувано можливість інтеграцій тренажеру до системи дистанційного навчання Moodle, на якій власне розміщуються дистанційні курси.

Розглянемо основні завдання роботи:

- проведено вибір та обгрунтування мови програмування;
- розроблено логічну схему інтерфейсу тренажеру з теми «Обрізання основ»;
- розглянуто теоретичні відомості та специфіку застосування обрізання основ;
- розроблено алгоритм тренажера з теми «Обрізання основ»;
- розроблено навчальний тренажер з даної теми.

Тренажер розроблено на основі двох прикладів. Наведено і теоретичну частину, і практичну для кожної задачі.

По завершенню тренажера виводиться шкала оцінювання. Після чого користувач може завершити тестування та перейти до теорії, або вийти. Також може розпочати тестування знову, або приступити до іншого завдання.

Тренажер готовий до використання в дистанційному курсі «Теорія програмування».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ємець О. О. Методичні рекомендації до виконання бакалаврської роботи для студентівза освітньою програмою «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки та інформаційні технології» галузь знань - 12 «Інформаційні технології» / О.О.(Олег) Ємець. – Полтава : РВВ ПУЕТ, 2017. – 71 с.
2. Дистанційний курс «Теорія програмування» // Головний центр дистанційного навчання вищого навчального закладу УКООПСПІЛКИ «Полтавський університет економіки і торгівлі».
3. Черненко О.О. Електронний навчально-методичний посібник для самостійного вивчення навчальної дисципліни «Теорія програмування» для студентів напряму 6.040302 «Інформатика».
4. Козлакова Г. О. Використання засобів Інтернет у науково-педагонічних дослідженнях. Нові інформаційні технології в начальних закладах України / Г. О. Козлакова // Матеріали Міжнародної конференції пам'яті проф. І. І. Мархеля . – Одеса, 2005.
5. Кравець В.О. Меморандум створення інформаційної освітньої мережі "Українська дистанційна освіта" / В.О. Кравець, В.М. Кухаренко . – Режим доступу: http://www.osvita.org.ua/distance/ukraine/add/00/
6. Абстрактне синтаксичне дерево [Електронний ресурс]– Режим доступу: https://uk.wikipedia.org/wiki/ . – 02 лютого 2016 р.
7. Дерево розбору. Перетворення дерева розбору в дерево операцій. [Електронний ресурс] – Режим доступу: https://studfile.net/preview/5465773/page:14/ - 01 листопада 2019
8. Бабій М.С. Теорія програмування: Навчальний посібник [Електронний ресурс] / М.С. Бабій, О.П. Чекалов.– Суми: Вид-во СумДУ, 2009. – 181 с.
9. Нікітченко М.С. Теоретичні основи програмування: Навчальний посібник [Електронний ресурс] / М.С. Нікітченко. – Київ: КНУ ім. Т.Г. Шевченка, 2009. – 200 с. – Режим доступу: http://ttp.unicyb.kiev.ua/doc/TOP.pdf.

10. Java [Електронний ресурс] // Матеріал з Вікіпедії — вільної енциклопедії. – Режим доступу: http://uk.wikipedia.org/wiki/Java.
11. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: ДСТУ 7.1-2006. – [Чинний від 2007-07-01]. – К. : Держспоживстандарт України, 2007. – 47 с.

ДОДАТОК А. КОД ПРОГРАМИ