ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ БІЗНЕСУ ТА СУЧАСНИХ ТЕХНОЛОГІЙ

ФОРМА НАВЧАННЯ ЗАОЧНА
КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту

Завідувач кафедри _____О.О. Ємець
(підпис)

«_____»_____2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ТРЕНАЖЕРА З ТЕМИ «ЛЯМБДА-ВИРАХУВАННЯ І ТИПІЗАЦІЯ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ТЕОРІЯ ПРОГРАМУВАННЯ»

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

Виконавець роботи Закіров Ренат Раїфович
_____ «___»_____2021р.
(підпис)

Науковий керівник к.ф.-м.н., доц., Черненко Оксана Олексіївна
_____ «___»_____2021р.
(підпис)

ПОЛТАВА 2021 р.

РЕФЕРАТ

Записка: 68 с., 22 рис., 1 додаток, 12 джерел.

Предмет розробки – тренажер з теми «Лямбда-вирахування і типізація».

Мета роботи – розробка програмного забезпечення тренажера з теми «Лямбда-вирахування і типізація» дистанційного навчального курсу «Теорія програмування».

Методи, які були використані для розв'язування задачі – застосування математичних основ теорії програмування з теми «Лямбда-вирахування і типізація», мова програмування Java.

Згідно алгоритму на стартовій сторінці пропонується переглянути теоретичний матеріал за темою або завантажити його. Також виводиться наступна інформація:

- тема тренажера;
- назва дисципліни;
- прізвище, ініціали розробника;
- прізвище, ініціали керівника.

На кожному кроці виводиться завдання та варіанти відповіді, серед яких слід вибрати одну правильну. При неправильній відповіді відображається повідомлення про помилку, що вказує на вірну відповідь.

На останньому кроці відображається перелік питань, в якому вказано було вибрано правильну відповідь чи ні. Пропонується закрити тренажер або перейти на стартову сторінку, де можна розпочати тестування спочатку.

Ключові слова: ТРЕНАЖЕР, ЛЯМБДА-ВИРАХУВАННЯ, ТИПІЗАЦІЯ, ТЕОРІЯ ПРОГРАМУВАННЯ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, скорочень, символів
λ-числення	Лямбда-числення
тренажер	комп’ютерна програма, призначена для вивчення і закріплення різноманітних практичних навичок
Java	об'єктно-орієнтована мова програмування
JFrame	контейнер вищого рівня
JPanel	компонент-контейнер

ВСТУП

Технологія дистанційного навчання посилює роль методів активного пізнання. Реалізацію технології дистанційного навчання можна забезпечити шляхом розробки моделі використання віртуально-навчальних середовищ.

Впровадження у навчальний процес віртуально-навчальних середовищ і надання студентам та викладачам необхідних методичних рекомендацій щодо використання віртуально-навчальних середовищ у фаховій підготовці забезпечить просування за критеріями та рівнями якості застосування технології дистанційного навчання у фаховій підготовці фахівців та надасть змогу підвищити якість фахової підготовки.

Таким чином, на сьогоднішній день, виникає потреба розробки і запровадження у навчальний процес програм дистанційного навчання, що відповідають кращим світовим зразкам і забезпечують підготовку фахівців на високому професійному рівні. Використання мережі Internet дає можливість оперативного доступу до інформаційних ресурсів навчального закладу та можливість ефективної взаємодії «викладач-студент», як в on-line, так і в off- line режимах.

Метою роботи є розробка програмного забезпечення тренажера з теми «Лямбда-вирахування і типізація» дистанційного навчального курсу «Теорія програмування».

Об’єктом розробки є процес навчання математичним дисциплінам.

Предмет розробки – тренажер з теми «Лямбда-вирахування і типізація».

Головне завдання – створити програмну реалізацію тренажера для закріплення знань.

Перелік використаних методів – застосування математичних основ теорії програмування з теми «Лямбда-вирахування і типізація», мова програмування Java.

Робота складається з чотирьох розділів. У першому розділі розглянуто постановку задачі. У другому розділі описано значення дистанційного навчання у ВНЗ, огляд робіт за схожою тематикою. У третьому розділі описано огляд матеріалу за темою роботи, алгоритмізацію задачі за темою роботи, розробку блок-схеми, яка підлягає програмуванню, обґрунтування вибору програмних засобів для реалізації завдання роботи. У четвертому розділі – вказано опис процесу програмної реалізації, опис програми, необхідну користувачу програми інструкцію.

Обсяг пояснювальної записки: 68 стор., в т.ч. основна частина - 49 стор., джерела - 12 назв.

1. ПОСТАНОВКА ЗАДАЧІ

Основним завданням роботи є розробка алгоритму тренажера з теми «Лямбда-вирахування і типізація». Розроблений тренажер за цим алгоритмом в подальшому буде використовуватися як складова дистанційного курсу «Теорія програмування».

Розглянемо основні завдання роботи:

- описати основні вимоги до тренажера;
- розглянути актуальність використання дистанційного навчання;
- навести теоретичний матеріал з теми для його використання в тренажері;
- розробити алгоритм тренажера;
- скласти блок-схему розробленого алгоритму;
- описати причини вибору мови програмування для реалізації тренажера;
- описати процес реалізації тренажера;
- розробити тренажер з даної теми.

На стартовій сторінці пропонується виведення наступної інформації:

- тема тренажера;
- назва дисципліни;
- прізвище, ініціали розробника;
- прізвище, ініціали керівника.

Слід надати можливість переглянути теоретичний матеріал або завантажити його в будь-який момент.

Тренажер представляє собою проходження тестування. На кожному кроці мають виводитися завдання та варіанти відповіді, серед яких має бути хоча б одна правильна. При неправильній відповіді обов’язково має відображатися повідомлення про помилку.

Після проходження всіх тестів потрібно виводити список завдань з вказаними відповідями, тобто, вказати де користувач відповів правильну, а де – ні. Також необхідно надати можливість закрити тренажер або знову розпочати тестування.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Дистанційне навчання у ВНЗ

Освітні навчальні заклади з традиційною системою навчання вже не в змозі задовольнити попит населення в освітніх послугах. Люди хочуть отримувати якісну освіту без відриву від виробництва, у зручний для себе час, у потрібній галузі знань. Однією із таких освітніх технологій, що здатна задовольнити зростаючий попит людей на якісні знання, забезпечити можливість «навчатись упродовж усього життя» – дистанційне навчання. Воно знайшло широке застосування в світі.

Дистанційне навчання у ВНЗ не є різновидом або вдосконаленим варіантом заочного, оскільки це нова, самостійна, прогресивна форма навчання, яка володіє більшими потенційними можливостями. Сфера можливого застосування дистанційного навчання у ВНЗ досить широка: від цілісної підготовки фахівців до окремих курсів та фрагментів дидактичного забезпечення під час різних видів занять. На основі аналізу світових тенденцій розвитку дистанційної освіти вважаємо за доцільне виокремити такі тенденції розвитку дистанційної освіти в Україні: інформаційно-технологічну, нормативно-правову, фінансово-економічну, інституційну, науково-методичну та організаційно - педагогічну.

Інформаційно-технологічна тенденція розвитку дистанційної освіти в Україні тісно пов’язана з інформатизацією, яка на сучасному історичному етапі має стратегічне значення для нашої країни. У наш час виробництво інформаційного продукту через його високу товарну вартість є важливим чинником економічного розвитку країни. Інформаційні технології, проникаючи в усі галузі діяльності людини, змінюють характер праці як у виробничій, так і у невиробничій сферах, впливають на структуру національної економіки, підвищують рівень інформування широких верств населення й таким чином сприяють демократизації суспільства. Ці

тенденції, що в усьому світі визначаються як процес інформатизації, позначаються на житті суспільства. Порівняно з традиційними індустріальними методами їх застосування дає можливість забезпечити підвищення рівня матеріалізації інтелектуальної праці і якості виробів. Саме через це найбільш розвинуті країни світу отримують на міжнародних ринках значні переваги.

На відміну від зарубіжних моделей, українська дистанційна освіта більш наближена до нашого споживача і є більш демократична. Органічно поєднуючи в собі змішані технології відкритої освіти (кейс-технології, TV-технології, мережеві технології), українська дистанційна освіта стає найбільш доступна широким масам населення, роблячи можливим здобувати освіту не на все життя, а все життя.

Сучасне інформаційне суспільство висуває вимоги до системи освіти, основні з яких можна сформулювати так: вміння самостійно знаходити, накопичувати і переосмислювати наукові знання; вміння студентів самостійно орієнтуватися в сучасному інформаційному суспільстві [2, 3].

Дистанційна освіта розвивається дуже швидко, і для України є перспективною формою вищої освіти. На Заході ця форма з’явилася вже досить давно і має велику популярність серед студентів через її економічні показники і навчальну ефективність. Дистанційну форму навчання ще називають «освітою на протязі всього життя» через те, що більшість тих, хто навчається – дорослі люди. Багато хто з них вже має вищу освіту, проте через необхідність підвищення кваліфікації або розширення сфери діяльності у багатьох виникає потреба швидко і якісно засвоїти нові знання і набути навички роботи. Саме тоді оптимальною формою може стати дистанційне навчання.

Для вдосконалення та поширення високих дистанційних технологій необхідне рішення двох основних проблем. Головна, знаходиться в області права, інша – в сфері фінансування робіт з розробки та впровадження

інноваційних технологій. З метою їх успішного вирішення об'єктивно необхідна реалізація наступних першочергових заходів і напрямів:

- розробка і реалізація Загальноукраїнської програми дистанційної безперервної освіти;
- викорінення протиріч в законодавстві про освіту в Україні, приведення його у відповідність з об'єктивними потребами і тенденціями розвитку дистанційних форм навчання;
- розробка наукових основ, що забезпечують інноваційність і дистанційних форм, і рівнів освіти, програм та навчальних планів;
- наукове обґрунтування ринку навчальної літератури, комп'ютерних та мультимедійних баз даних, виключення можливості його монополізації;
- створення варіативних методик з дистанційного навчання людей з різними рівнями здібностей, віком і потребами;
- забезпечення переходу до інтерактивних методів та практичної спрямованості дистанційного навчання;
- створення системи підтримки проектів, нововведень в технології дистанційної освіти, її заочних та інших форм;
- надання права навчання студентів, отримання атестатів і дипломів у різних освітніх закладах.

Відкрита освіта дає широке поле для наукових досліджень, що сприяють розвитку творчих ініціатив розробників і педагогів, переходячи з об'єкта вивчення в об'єкт прогнозування, конструювання та проектування [2, 4].

Впровадження дистанційної освіти в Україні здійснювався за умов досить низького рівня інформатизації українського суспільства та розробки спеціалізованих методик дистанційного навчання. Кількість наукових організацій та вищих навчальних закладів України, які активно розробляють або використовують відповідні курси дистанційного навчання, досить незначна. Застосування технології дистанційного

навчання у фаховій підготовці майбутніх фахівців в університеті потребує змін у методиці викладання. Викладач поступово перестає бути для студента єдиним джерелом отримання знань. Студент багато інформації отримує в мережі Інтернет та за її допомогою.

Крім цього, система дистанційного навчання розрахована, в основному, на людей достатньо свідомих, які не потребують постійного контролю з боку викладача, тому важливу роль у цьому випадку відіграє мотивація студентів, їх здатність до самоорганізації. Якщо за традиційних форм навчання основною задачею студента було запам’ятати матеріал та потім його відтворити, то за умови застосування дистанційних технологій у студентів розвиваються уміння співставлення, синтезу, аналізу, оцінювання виявлення зв’язків, планування, групової взаємодії з використанням інформаційно-комунікаційних технологій та технології дистанційного навчання.

2.2. Огляд робіт за схожою тематикою

Розглянувши статті в матеріалах конференцій, можна виявити як потрібно розробляти алгоритм роботи тренажера і вже по ньому реалізувати саму програму.

Так, в статті Кильника В.В. пропонуються такий процес проходження:

1. Завантаження тренажера. Відображення автора. Можливість почати тренування чи перейти до теоретичних відомостей. Якщо користувач обирає «Розпочати тренування» перехід на наступний крок.

2. Відображення теоретичного запитання, варіантів відповідей. Якщо користувач натискає «Далі», збереження відповідей в пам’яті, перехід на наступний крок.

3. Після проходження теоретичних запитань, відображення практичного завдання.

Якщо користувач натискає «Далі», виконуємо перевірку заповнених коефіцієнтів, при помилці пропонуємо користувачеві повторити чи скористатися допомогою.

Якщо помилок не виявлено тоді перехід на крок 4.

Якщо користувач обирає «Повторити» то відновлюємо форму, перехід на крок 3.

Якщо обирає «Допомога» тоді відображаємо інформацію яким чином потрібно заповнювати коефіцієнти. Перехід на крок 3.

4. Після завершення всіх кроків відображаємо інформацію про проходження з демонстрацією набраних балів та кількості помилок.

Якщо користувач обирає «Повторити тренування», тоді перехід на крок 2.

Якщо обрано «Завершити» - завершуємо роботу тренажера [5].

Цікавим прикладом роботи є поступове заповнення таблиці, розрахувавши значення параметрів послідовно, що описано в статті Задорожнього А.В.

На певному етапі користувачу відображується робоче вікно тренажера. В ньому міститься умова завдання. За заданою вибіркою (табл.2.1) проводиться аналіз залежності витрат на відпустку Y (тис.у.о.) домогосподарства від кількості членів родини X. Також визначається вид залежності, оцінюються параметри рівняння регресії, сила лінійної залежності між X та Y. [5].

Таблиця 2.1. – Задана вибірка

№ п/п	*Кількість членів родини* x_i	*Витрти на відпустку (тис. у.о.)* y_i	x_iy_i	x_i^2	*ур*
1	**2**	**3**	**4**		**6**
1	1	16			
2	2	12			
3	5	23			
4	4	19			
5	6	30			
Всього ∑					
n =					
b1 =					
b0 =					

В іншій статті Задорожнього А.В. пропонується структуру інтерфейсу типових сторінок тренажера .

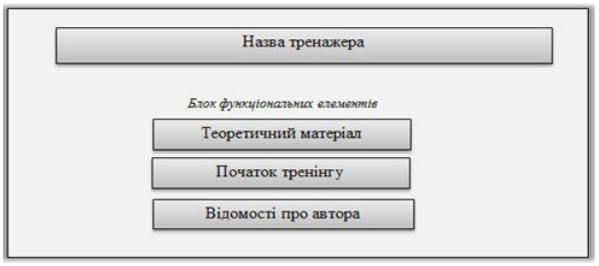

Рисунок 2.1 – Логічна структура титульної сторінки тренажера

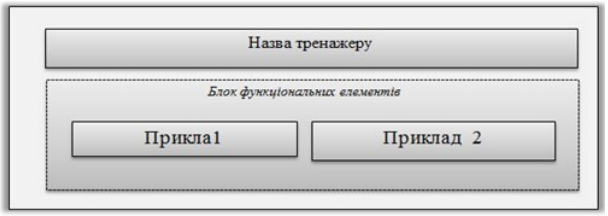

Рисунок 2.2 – Логічна структура сторінки вибору задачі

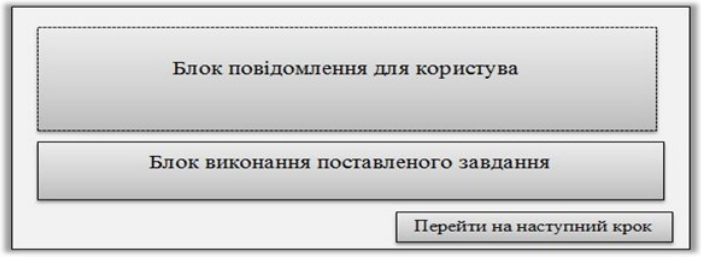

Рисунок 2.3 – Логічна структура інтерфейсу сторінки тренінгу

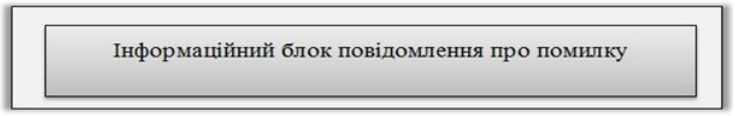

Рисунок 2.4 – Логічна структура інтерфейсу блоку повідомлення про помилку

Для програмної реалізації поставленої задачі важливим етапом є розробка алгоритму тренажера – покрокового виконання дій користувачем [6].

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Огляд матеріалу за темою роботи

Теорія типів — це будь-яка формальна система, що може слугувати альтернативою наївній теорії множин. У теорії типів кожен «терм» має «тип», а операції виконуються в узгодженості з цими типами.

Теорія типів тісно пов'язана з системами типів — особливістю мов програмування, яка призначена для зменшення кількості помилок. Теорія типів була розроблена для обходження парадоксів формальної логіки.

Однією з найвідоміших теорій типів є типізоване лямбда-числення Алонзо Черча та інтуїціоністська теорія типів Пера Мартіна-Льофа.

Лямбда-числення (λ-числення) - формальна система, розроблена американським математиком Алонзо Черчем для формалізації та аналізу поняття обчислюваності.

Типізоване лямбда-числення — це система лямбда-числення, у якій кожний вислів має тип. У цьому контексті тип — це формула певної системи числення, як-от (інтуїціоністської) пропозиційної логіки або логіки першого порядку. Типізоване лямбда-числення з простими типами (де типи — це формули імплікаційного фрагменту інтуїціоністської логіки) має застосування на практиці в функціональних мовах програмування, як от Haskell. Складніші типізовані системи є важливими для вивчення загальних характеристик обчислюваності та у сфері автоматиції автоматичних доведень.

Чисте λ-числення, терми якого, звані також об'єктами («обами»), або λ-термами, побудовані виключно з змінних застосуванням аплікації і абстракції. Спочатку наявність будь-яких констант не передбачається.

В основу λ-числення покладені дві фундаментальні операції:

- Аплікація (лат. Applicatio - прикладання, приєднання) означає застосування або виклик функції по відношенню до заданого значення. Її

зазвичай позначають $f\ a$, де f - функція, а a - аргумент. Це відповідає загальноприйнятому в математиці запису $f\ (a)$, який теж іноді використовується, однак для λ-числення важливо те, що f трактується як алгоритм, що обчислює результат по заданому вхідному значенню. У цьому сенсі аплікація f до a може розглядатися двояко: як результат застосування f до a, або ж як процес обчислення $f\ a$. Остання інтерпретація аплікації пов'язана з поняттям β-редукції.

- Абстракція або λ-абстракція (лат. Abstractio - відволікання, відділення) в свою чергу будує функції по заданим виразам. Саме, $t \equiv t[x]$- вираз, що вільно містить x, тоді запис $\lambda x.t[x]$ означає: λ функція від аргументу x, яка має вигляд $t[x]$, позначає функцію $x \mapsto t[x]$. Таким чином, за допомогою абстракції можна конструювати нові функції. Вимога, щоб x вільно входив в t, що не дуже істотно - досить припустити, що $\lambda x.t \equiv t$, якщо це не так [7,8].

α-еквівалентність.

Основна форма еквівалентності, яка визначається в лямбда-термах, це альфа-еквівалентність. Наприклад, $\lambda x.x$ і $\lambda y.y$: альфа-еквівалентні лямбда-терми і обидва представляють одну і ту ж функцію (функцію тотожності). Терми x і y не альфа-еквівалентні, так як вони не знаходяться в лямбда-абстракції.

β-редукція.

Оскільки вираз $\lambda x.2 \cdot x + 1$ позначає функцію, яка ставить у відповідність кожному x значення $2 \cdot x + 1$, то для обчислення виразу

$$(\lambda x.2 \cdot x + 1)3,$$

в яке входять і аплікація і абстракція, необхідно виконати підстановку числа 3 в терм $2 \cdot x + 1$ замість змінної x. В результаті виходить $2 \cdot 3 + 1 = 7$. Це міркування в загальному вигляді записується як

$$(\lambda x.t)a = t[x := a],$$

і носить назву β-редукція.

Вираз виду $(\lambda x.t)a$, тобто застосування абстракції до деякого терму, називається редексом (redex). Незважаючи на те, що β-редукція по суті є єдиною «істотною» аксіомою λ-числення, вона призводить до вельми змістовної і складної теорії. Разом з нею λ-числення має властивість повноти по Тьюрингу і, отже, являє собою найпростішу мову програмування.

η-перетворення.

η-перетворення висловлює ту ідею, що дві функції є ідентичними тоді і тільки тоді, коли, будучи застосованими до будь-якого аргументу, дають однакові результати. η-перетворення переводить один в одного формули $\lambda x.f$ і f (тільки якщо x не має вільних входжень в f: інакше, вільна змінна x після перетворення стане зв'язаною зовнішньою абстракцією або навпаки).

Каррінг.

Функція двох змінних x і y $f(x,y)=x+y$ може бути розглянута як функція однієї змінної x, яка повертає функцію однієї змінної y, тобто як вираження $\lambda x.\lambda y.x+y$. Такий прийом працює точно так же для функцій будь-якої арності. Це показує, що функції багатьох змінних можуть бути виражені в λ-численні і є «синтаксичним цукром». Описаний процес перетворення функцій багатьох змінних в функцію однієї змінної називається каррінг (також: каррінг), в честь американського математика Хаскелл Каррі, хоча першим його запропонував М. Е. Шейнфінкель (1924).

Семантика безтипового λ-числення.

Той факт, що терми λ-числення діють як функції, що застосовуються до термів λ-числення (тобто, можливо, до самих себе), призводить до складнощів побудови адекватної семантики λ-числення. Щоб надати λ-численню будь-який сенс, необхідно отримати множину D, в яку вкладався б його простір функцій $D \to D$. У загальному випадку такого D

не існує з міркувань обмежень на потужності цих двох множин, D і функцій з D в D: друга має більшу потужність, ніж перша.

Цю трудність на початку 1970-х років подолав Дана Скотт, побудувавши поняття області D (спочатку на повних гратах, в подальшому узагальнивши до повної частково-впорядкованої множини зі спеціальною топологією) і урізавши $D \to D$ кілька разів безперервних в цій топології функцій. На основі цих побудов була створена денотаціонна семантика мов програмування, зокрема, завдяки тому, що за допомогою них можна додати точний сенс таким двом важливим конструкціям мов програмування, як рекурсія і типи даних.

Зв'язок з рекурсивними функціями.

Рекурсія - це визначення функції через себе; на перший погляд, лямбда-числення не дозволяє цього, але це враження оманливе. Наприклад, розглянемо рекурсивну функцію, яка обчислює факторіал:

f(n) = 1, if n = 0; else n × f(n - 1).

В лямбда-численні функція не може безпосередньо посилатися на себе. Проте, функції може бути переданий параметр, пов'язаний з нею. Як правило, цей аргумент стоїть на першому місці. Зв'язавши його з функцією, ми отримуємо нову, вже рекурсивну функцію. Для цього аргумент, який посилається на себе (тут позначений як *r*), обов'язково повинен бути переданий в тіло функції.

g := λr. λn.(1, if n = 0; else n × (r r (n-1)))

f := g g

Це вирішує специфічну проблему обчислення факторіала, але рішення в загальному вигляді також можливо. Отримавши лямбда-терм, що представляє тіло рекурсивної функції або цикл, передавши себе в якості першого аргументу, комбінатор нерухомої точки поверне необхідну рекурсивну функцію або цикл. Функції не потребують явної передачі себе кожен раз.

Існує кілька визначень комбінаторів нерухомої точки. Найпростіший з них:

Y = λg.(λx.g (x x)) (λx.g (x x))

В лямбда-численні, $Y\,g$- нерухома точка g; продемонструємо це:

Y g

(λh.(λx.h (x x)) (λx.h (x x))) g

(λx.g (x x)) (λx.g (x x))

g ((λx.g (x x)) (λx.g (x x)))

g (Y g)

Тепер, щоб визначити факторіал, як рекурсивну функцію, ми можемо просто написати $g\left(Y\,g\right)n$, де n - число, для якого обчислюється факторіал. Нехай n = 4, отримуємо:

g (Y g) 4

(λfn.(1, if n = 0; and n·(f(n-1)), if n>0)) (Y g) 4

(λn.(1, if n = 0; and n·((Y g) (n-1)), if n>0)) 4

1, if 4 = 0; and 4·(g(Y g) (4-1)), if 4>0

4·(g(Y g) 3)

4·(λn.(1, if n = 0; and n·((Y g) (n-1)), if n>0) 3)

4·(1, if 3 = 0; and 3·(g(Y g) (3-1)), if 3>0)

4·(3·(g(Y g) 2))

4·(3·(λn.(1, if n = 0; and n·((Y g) (n-1)), if n>0) 2))

4·(3·(1, if 2 = 0; and 2·(g(Y g) (2-1)), if 2>0))

4·(3·(2·(g(Y g) 1)))

4·(3·(2·(λn.(1, if n = 0; and n·((Y g) (n-1)), if n>0) 1)))

4·(3·(2·(1, if 1 = 0; and 1·((Y g) (1-1)), if 1>0)))

4·(3·(2·(1·((Y g) 0))))

4·(3·(2·(1·((λn.(1, if n = 0; and n·((Y g) (n-1)), if n>0) 0))))

4·(3·(2·(1·(1, if 0 = 0; and 0·((Y g) (0-1)), if 0>0))))

4·(3·(2·(1·(1))))

24

Кожне визначення рекурсивної функції може бути представлено як нерухома точка відповідної функції, отже, використовуючи *Y*, кожне рекурсивне визначення може бути виражено як лямбда-вираз. Зокрема, ми можемо визначити віднімання, множення, порівняння натуральних чисел рекурсивно.

В мовах програмування.

У мовах програмування під «λ-обчисленням» часто розуміється механізм «анонімних функцій» - callback-функцій, які можна визначити прямо в тому місці, де вони використовуються, і які мають доступ до локальних змінних поточної функції (замикання) [9].

Просто типізоване лямбда-числення (просте типізоване лямбда-числення, лямбда-числення з простими типами, система $\lambda^{\rightarrow}$) - система типізованого лямбда-числення, в якому лямбда абстракції приписується спеціальний «стрілочний» тип. Ця система була запропонована Алонзо Черчем в 1940 році. Для близького до лямбда-числення формалізму комбінаторної логіки схожа система розглядалася Хаскелл Каррі в 1934 році.

У базовій версії системи $\lambda^{\rightarrow}$ типи конструюються з набору змінних за допомогою єдиного бінарного інфіксного конструктора →. За традицією для змінних типу використовують грецькі літери, а → вважають правоассоціативним, $\alpha \to \beta \to \gamma$ є скороченням для $\alpha \to (\beta \to \gamma)$. Букви з другої половини грецького алфавіту (σ, τ, і т.д.) часто використовуються для позначення довільних типів, а не тільки змінних типу.

Розрізняють дві версії просто типізованої системи. Якщо в якості термів використовуються ті ж терми, що і в безтиповому лямбда-численні, то систему називають неявно типізованою або типізованою по Каррі. Якщо ж змінні в лямбда-абстракції анотуються типами, то систему називають явно типізованою або типізованою по Черчу. Як приклад наведемо тотожну функцію в стилі Каррі: $\lambda x.x$, і в стилі Черча: $\lambda x:\alpha.x$.

Правила редукції не відрізняються від правил для безтипового лямбда-числення. β-редукція визначається через підстановку

$$(\lambda x:\sigma.M)N \to_\beta M[x:=N]$$

η-редукція визначається так

$$\lambda x:\sigma.Mx \to_\eta M$$

Для η-редукції потрібно, щоб змінна x була вільною в термі M.

Контекстом називається множина тверджень про типізації змінних, розділених комою.

Контексти зазвичай позначають великими грецькими буквами: Γ, Δ. У контекст можна додати «свіжу» для цього контексту змінну: якщо Γ - допустимий контекст, який не містить змінної x, то $\Gamma, x:\alpha$ - теж допустимий контекст.

Загальний вигляд твердження про типізації такий:

$$\Gamma \vdash M:\sigma$$

Це читається таким чином: в контексті Γ терм M має тип σ.

У просто типізованому лямбда-численні приписування типів термам здійснюється за наведеними нижче правилами.

Аксіома. Якщо змінній x присвоєно в контексті тип σ, то в цьому контексті x має тип σ. У вигляді правила виводу:

$$\frac{x:\sigma \in \Gamma}{\Gamma \vdash x:\sigma}$$

Правило введення →. Якщо в деякому контексті, розширеному твердженням, що x має тип σ, терм M має тип τ, то в згаданому контексті (без x), лямбда-абстракція $\lambda x:\sigma.M$ має тип $\sigma \to \tau$. У вигляді правила виводу:

$$\frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash (\lambda x:\sigma.M):(\sigma \to \tau)}$$

Правило видалення →. Якщо в деякому контексті терм M має тип $\sigma \to \tau$, а терм N має тип σ, то застосування $M\ N$ має тип τ. У вигляді правила виводу:

$$\frac{\Gamma \vdash M : (\sigma \to \tau)\ \Gamma \vdash N : \sigma}{\Gamma \vdash (MN) : \tau}$$

Перше правило дозволяє приписати тип вільним змінним, задавши їх у контексті. Друге правило дозволяє типізувати лямбда-абстракцію стрілочним типом, прибираючи з контексту змінну, що зв'язана цієї абстракцією. Третє правило дозволяє типізувати аплікацію (застосування) за умови, що лівий аплікант має відповідний стрілочний тип.

Приклади тверджень про типізації в стилі Черча:

$x : \alpha \vdash x : \alpha$ (аксіома)

$\vdash (\lambda x : \alpha . x) : (\alpha \to \alpha)$ (введення →)

$f : (\beta \to \gamma \to \delta), y : \beta \vdash (fy) : (\gamma \to \delta)$ (видалення →) [10]

3.2. Алгоритмізація задачі за темою роботи

На стартовій сторінці пропонується переглянути теоретичний матеріал за темою або завантажити його. Також виводиться наступна інформація:

- тема тренажера;
- назва дисципліни;
- прізвище, ініціали розробника;
- прізвище, ініціали керівника.

На кожному кроці виводиться завдання та варіанти відповіді, серед яких слід вибрати одну правильну. При неправильній відповіді відображається повідомлення про помилку, що вказує на вірну відповідь.

Крок 1. Теорія типів — це

Виберіть правильну відповідь:

- будь-яка формальна система, що може слугувати альтернативою наївній теорії множин. У теорії типів кожен «терм» має «тип», а операції виконуються в узгодженості з цими типами.
- це система лямбда-числення, у якій кожний вислів має тип. У цьому контексті тип — це формула певної системи числення, як-от (інтуїціоністської) пропозиційної логіки або логіки першого порядку.
- формальна система, розроблена американським математиком Алонзо Черчем для формалізації та аналізу поняття обчислюваності.

Якщо відповідь – перший варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Теорія типів — це будь-яка формальна система, що може слугувати альтернативою наївній теорії множин. У теорії типів кожен «терм» має «тип», а операції виконуються в узгодженості з цими типами».

Крок 2. Лямбда-числення (λ-числення) — це

Виберіть правильну відповідь:

- будь-яка формальна система, що може слугувати альтернативою наївній теорії множин. У теорії типів кожен «терм» має «тип», а операції виконуються в узгодженості з цими типами.
- це система лямбда-числення, у якій кожний вислів має тип. У цьому контексті тип — це формула певної системи числення, як-от (інтуїціоністської) пропозиційної логіки або логіки першого порядку.
- формальна система, розроблена американським математиком Алонзо Черчем для формалізації та аналізу поняття обчислюваності.

Якщо відповідь – третій варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Лямбда-числення (λ-числення) - формальна система, розроблена американським математиком Алонзо Черчем для формалізації та аналізу поняття обчислюваності».

Крок 3. Типізоване лямбда-числення — це

Виберіть правильну відповідь:

- система типізованого лямбда-числення, в якому лямбда абстракції приписується спеціальний «стрілочний» тип.
- це система лямбда-числення, у якій кожний вислів має тип. У цьому контексті тип — це формула певної системи числення, як-от (інтуїціоністської) пропозиційної логіки або логіки першого порядку.
- формальна система, розроблена американським математиком Алонзо Черчем для формалізації та аналізу поняття обчислюваності.

Якщо відповідь – другий варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Типізоване лямбда-числення — це система лямбда-числення, у якій кожний вислів має тип. У цьому контексті тип — це формула певної системи числення, як-от (інтуїціоністської) пропозиційної логіки або логіки першого порядку».

Крок 4. В основу λ-числення покладені дві фундаментальні операції. Аплікація (лат. Applicatio - прикладання, приєднання) означає

Виберіть правильну відповідь:

- побудову функції по заданим виразам. Саме, $t \equiv t[x]$- вираз, що вільно містить x, тоді запис $\lambda x.t[x]$ означає: λ функція від аргументу x, яка має вигляд $t[x]$, позначає функцію $x \mapsto t[x]$.

- застосування або виклик функції по відношенню до заданого значення. Її зазвичай позначають $f\ a$, де f - функція, а a - аргумент.
- лише застосування функції по відношенню до заданого значення.

Якщо відповідь – другий варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Аплікація (лат. Applicatio - прикладання, приєднання) означає застосування або виклик функції по відношенню до заданого значення. Її зазвичай позначають $f\ a$, де f - функція, а a - аргумент».

Крок 5. В основу λ-числення покладені дві фундаментальні операції. Абстракція або λ-абстракція (лат. Abstractio - відволікання, відділення) в свою чергу

Виберіть правильну відповідь:

- будує функції по заданим виразам. Саме, $t \equiv t[x]$- вираз, що вільно містить x, тоді запис $\lambda x.t[x]$ означає: λ функція від аргументу x, яка має вигляд $t[x]$, позначає функцію $x \mapsto t[x]$.
- означає застосування або виклик функції по відношенню до заданого значення. Її зазвичай позначають $f\ a$, де f - функція, а a - аргумент.
- означає лише застосування функції по відношенню до заданого значення.

Якщо відповідь – перший варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Абстракція або λ-абстракція (лат. Abstractio - відволікання, відділення) в свою чергу будує функції по заданим виразам. Саме, $t \equiv t[x]$- вираз, що вільно містить x, тоді запис $\lambda x.t[x]$ означає: λ функція від аргументу x, яка має вигляд $t[x]$, позначає функцію $x \mapsto t[x]$».

Крок 6. Основна форма еквівалентності, яка визначається в лямбда-термах, це альфа-еквівалентність. Терми *x* і *y* не альфа-еквівалентні, так як вони не знаходяться в лямбда-абстракції.

Введіть правильну відповідь:

- Наприклад, $\lambda x.$____ і $\lambda y.$____: альфа-еквівалентні лямбда-терми і обидва представляють одну і ту ж функцію (функцію тотожності).

Якщо відповідь – $\lambda x.x$ і $\lambda y.y$, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Наприклад, $\lambda x.x$ і $\lambda y.y$: альфа-еквівалентні лямбда-терми і обидва представляють одну і ту ж функцію (функцію тотожності).».

Крок 7. Оскільки вираз $\lambda x.2 \cdot x + 1$ позначає функцію, яка ставить у відповідність кожному *x* значення $2 \cdot x + 1$, то для обчислення виразу $(\lambda x.2 \cdot x + 1)3$, в яке входять і аплікація і абстракція, необхідно виконати підстановку числа 3 в терм $2 \cdot x + 1$ замість змінної *x*. В результаті виходить $2 \cdot 3 + 1 = 7$. Це міркування носить назву β-редукція і в загальному вигляді записується як

Введіть правильну відповідь:

- $(\lambda x.t)a =$ ____

Якщо відповідь – $(\lambda x.t)a = t[x := a]$, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Це міркування в загальному вигляді записується як $(\lambda x.t)a = t[x := a]$, і носить назву β-редукція».

Крок 8. η-перетворення висловлює ту ідею, що дві функції є ідентичними тоді і тільки тоді, коли, будучи застосованими до будь-якого аргументу, дають однакові результати. η-перетворення переводить один в одного формули

Введіть правильну відповідь:

- $\lambda x.f$ і ____

Якщо відповідь – $\lambda x.f$ і f, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «η-перетворення переводить один в одного формули $\lambda x.f$ і f».

Крок 9. Процес перетворення функцій багатьох змінних в функцію однієї змінної називається каррінг. Функція двох змінних x і y $f(x,y)=x+y$ може бути розглянута як функція однієї змінної x, яка повертає функцію однієї змінної y, тобто як вираження

Введіть правильну відповідь:

- $\lambda x.\lambda y.$_____

Якщо відповідь – $\lambda x.\lambda y.x+y$, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Функція двох змінних x і y $f(x,y)=x+y$ може бути розглянута як функція однієї змінної x, яка повертає функцію однієї змінної y, тобто як вираження $\lambda x.\lambda y.x+y$.».

Крок 10. Просто типізоване лямбда-числення — це

Виберіть правильну відповідь:

- система типізованого лямбда-числення, в якому лямбда абстракції приписується спеціальний «стрілочний» тип.
- це система лямбда-числення, у якій кожний вислів має тип. У цьому контексті тип — це формула певної системи числення, як-от (інтуїціоністської) пропозиційної логіки або логіки першого порядку.
- формальна система, розроблена американським математиком Алонзо Черчем для формалізації та аналізу поняття обчислюваності.

Якщо відповідь – перший варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Просто типізоване лямбда-числення (просте типізоване лямбда-числення, лямбда-числення з простими типами, система $\lambda^{\rightarrow}$) - система типізованого лямбда-

числення, в якому лямбда абстракції приписується спеціальний «стрілочний» тип».

Крок 11. У базовій версії системи $\lambda^{\rightarrow}$ типи конструюються з набору змінних за допомогою єдиного бінарного інфіксного конструктора $\rightarrow$. За традицією для змінних типу використовують

Виберіть правильну відповідь:

- англійські літери.
- латинські літери.
- грецькі літери.

Якщо відповідь – третій варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «За традицією для змінних типу використовують грецькі літери».

Крок 12. За традицією для змінних типу використовують грецькі літери, а $\rightarrow$ вважають

Виберіть правильну відповідь:

- лівоассоціативним.
- правоассоціативним.

Якщо відповідь – другий варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «За традицією для змінних типу використовують грецькі літери, а $\rightarrow$ вважають правоассоціативним».

Крок 13. Чи правильним є твердження, що $\alpha \rightarrow \beta \rightarrow \gamma$ є скороченням для $\alpha \rightarrow (\beta \rightarrow \gamma)$?

Виберіть правильну відповідь:

- так.
- ні.

Якщо відповідь – перший варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «$\alpha \rightarrow \beta \rightarrow \gamma$ є скороченням для $\alpha \rightarrow (\beta \rightarrow \gamma)$».

Крок 14. Букви з другої половини грецького алфавіту (σ, τ, і т.д.) часто використовуються для позначення

Виберіть правильну відповідь:

- довільних типів.
- змінних типу.
- довільних типів і змінних типу.

Якщо відповідь – третій варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Букви з другої половини грецького алфавіту (σ, τ, і т.д.) часто використовуються для позначення довільних типів, а не тільки змінних типу».

Крок 15. Розрізняють дві версії просто типізованої системи. Якщо в якості термів використовуються ті ж терми, що і в безтиповому лямбда-численні, то систему називають

Виберіть правильну відповідь:

- явно типізованою або типізованою по Черчу. Як приклад наведемо тотожну функцію в стилі Черча: $\lambda x : \alpha . x$.
- неявно типізованою або типізованою по Каррі. Як приклад наведемо тотожну функцію в стилі Каррі: $\lambda x . x$.
- просто типізованою.

Якщо відповідь – другий варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Якщо в якості термів використовуються ті ж терми, що і в безтиповому лямбда-численні, то систему називають неявно типізованою або типізованою по Каррі».

Крок 16. Якщо ж змінні в лямбда-абстракції анотуються типами, то систему називають

Виберіть правильну відповідь:

- явно типізованою або типізованою по Черчу. Як приклад наведемо тотожну функцію в стилі Черча: $\lambda x : \alpha . x$.

- неявно типізованою або типізованою по Каррі. Як приклад наведемо тотожну функцію в стилі Каррі: $\lambda x.x$.
- просто типізованою.

Якщо відповідь – перший варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Якщо ж змінні в лямбда-абстракції анотуються типами, то систему називають явно типізованою або типізованою по Черчу».

Крок 17. Правила редукції не відрізняються від правил для безтипового лямбда-числення. β-редукція визначається через підстановку

Виберіть правильну відповідь:

- $\Gamma \vdash M : \sigma$.
- $\lambda x : \sigma.Mx \to_{\eta} M$.
- $(\lambda x : \sigma.M)N \to_{\beta} M[x := N]$.

Якщо відповідь – третій варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «β-редукція визначається через підстановку

$$(\lambda x : \sigma.M)N \to_{\beta} M[x := N]».$$

Крок 18. Для η-редукції потрібно, щоб змінна *x* була вільною в термі *M*. η-редукція визначається так

Виберіть правильну відповідь:

- $\Gamma \vdash M : \sigma$.
- $\lambda x : \sigma.Mx \to_{\eta} M$.
- $(\lambda x : \sigma.M)N \to_{\beta} M[x := N]$.

Якщо відповідь – другий варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «η-редукція визначається так

$$\lambda x : \sigma.Mx \to_{\eta} M ».$$

Крок 19. Контекстом називається множина тверджень про типізації змінних, розділених комою. Контексти зазвичай позначають великими грецькими буквами: Γ, Δ. Загальний вигляд твердження про типізації такий:

Виберіть правильну відповідь:

- $\Gamma \vdash M : \sigma$.
- $\lambda x : \sigma . Mx \to_{\eta} M$.
- $(\lambda x : \sigma . M) N \to_{\beta} M[x := N]$.

Якщо відповідь – перший варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Загальний вигляд твердження про типізації такий:

$$\Gamma \vdash M : \sigma \text{».}$$

Крок 20. У просто типізованому лямбда-численні приписування типів термам здійснюється за наведеними нижче правилами. Аксіома –

Виберіть правильну відповідь:

- Якщо в деякому контексті, розширеному твердженням, що x має тип σ, терм M має тип τ, то в згаданому контексті (без x), лямбда-абстракція $\lambda x : \sigma . M$ має тип $\sigma \to \tau$.
- Якщо змінній x присвоєно в контексті тип σ, то в цьому контексті x має тип σ.
- Якщо в деякому контексті терм M має тип $\sigma \to \tau$, а терм N має тип σ, то застосування $M N$ має тип τ.

Якщо відповідь – другий варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Аксіома. Якщо змінній x присвоєно в контексті тип σ, то в цьому контексті x має тип σ».

Крок 21. Аксіома. Якщо змінній x присвоєно в контексті тип σ, то в цьому контексті x має тип σ. У вигляді правила виводу:

Виберіть правильну відповідь:

- $\dfrac{x:\sigma\in\Gamma}{\Gamma\vert\!-x:\sigma}$.
- $\dfrac{\Gamma,x:\sigma\vert\!-M:\tau}{\Gamma\vert\!-(\lambda x:\sigma.M):(\sigma\to\tau)}$.
- $\dfrac{\Gamma\vert\!-M:(\sigma\to\tau)\;\Gamma\vert\!-N:\sigma}{\Gamma\vert\!-(MN):\tau}$.

Якщо відповідь – перший варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «У вигляді правила виводу:

$$\frac{x:\sigma\in\Gamma}{\Gamma\vert\!-x:\sigma}\text{».}$$

Крок 22. Правило введення $\to$ –

Виберіть правильну відповідь:

- Якщо в деякому контексті, розширеному твердженням, що x має тип σ, терм M має тип τ, то в згаданому контексті (без x), лямбда-абстракція $\lambda x:\sigma.M$ має тип $\sigma\to\tau$.
- Якщо змінній x присвоєно в контексті тип σ, то в цьому контексті x має тип σ.
- Якщо в деякому контексті терм M має тип $\sigma\to\tau$, а терм N має тип σ, то застосування $M\,N$ має тип τ.

Якщо відповідь – перший варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Правило введення $\to$. Якщо в деякому контексті, розширеному твердженням, що x має тип σ, терм M має тип τ, то в згаданому контексті (без x), лямбда-абстракція $\lambda x:\sigma.M$ має тип $\sigma\to\tau$».

Крок 23. Правило введення $\to$. Якщо в деякому контексті, розширеному твердженням, що x має тип σ, терм M має тип τ, то в згаданому контексті (без x), лямбда-абстракція $\lambda x:\sigma.M$ має тип $\sigma\to\tau$. У вигляді правила виводу:

Виберіть правильну відповідь:

- $\dfrac{x:\sigma\in\Gamma}{\Gamma\vert\!-x:\sigma}$.
- $\dfrac{\Gamma,x:\sigma\vert\!-M:\tau}{\Gamma\vert\!-(\lambda x:\sigma.M):(\sigma\to\tau)}$.
- $\dfrac{\Gamma\vert\!-M:(\sigma\to\tau)\,\Gamma\vert\!-N:\sigma}{\Gamma\vert\!-(MN):\tau}$.

Якщо відповідь – другий варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «У вигляді правила виводу:

$$\frac{\Gamma,x:\sigma\vert\!-M:\tau}{\Gamma\vert\!-(\lambda x:\sigma.M):(\sigma\to\tau)}\text{».}$$

Крок 24. Правило видалення $\to$ –

Виберіть правильну відповідь:

- Якщо в деякому контексті, розширеному твердженням, що x має тип σ, терм M має тип τ, то в згаданому контексті (без x), лямбда-абстракція $\lambda x:\sigma.M$ має тип $\sigma\to\tau$.
- Якщо змінній x присвоєно в контексті тип σ, то в цьому контексті x має тип σ.
- Якщо в деякому контексті терм M має тип $\sigma\to\tau$, а терм N має тип σ, то застосування $M\,N$ має тип τ.

Якщо відповідь – третій варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Правило видалення $\to$. Якщо в деякому контексті терм M має тип $\sigma\to\tau$, а терм N має тип σ, то застосування $M\,N$ має тип τ».

Крок 25. Правило видалення $\to$. Якщо в деякому контексті терм M має тип $\sigma\to\tau$, а терм N має тип σ, то застосування $M\,N$ має тип τ. У вигляді правила виводу:

Виберіть правильну відповідь:

- $\dfrac{x:\sigma\in\Gamma}{\Gamma\vdash x:\sigma}$.
- $\dfrac{\Gamma,x:\sigma\vdash M:\tau}{\Gamma\vdash(\lambda x:\sigma.M):(\sigma\to\tau)}$.
- $\dfrac{\Gamma\vdash M:(\sigma\to\tau)\;\Gamma\vdash N:\sigma}{\Gamma\vdash(MN):\tau}$.

Якщо відповідь – третій варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «У вигляді правила виводу:

$$\frac{\Gamma\vdash M:(\sigma\to\tau)\;\Gamma\vdash N:\sigma}{\Gamma\vdash(MN):\tau}\text{».}$$

Крок 26. Перше правило дозволяє приписати тип вільним змінним, задавши їх у контексті.

Виберіть правильну відповідь:

- Приклад, $x:\alpha\vdash x:\alpha$
- Приклад, $f:(\beta\to\gamma\to\delta),y:\beta\vdash(fy):(\gamma\to\delta)$
- Приклад, $\vdash(\lambda x:\alpha.x):(\alpha\to\alpha)$

Якщо відповідь – перший варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Приклад, $x:\alpha\vdash x:\alpha$».

Крок 27. Друге правило дозволяє типізувати лямбда-абстракцію стрілочним типом, прибираючи з контексту змінну, що зв'язана цієї абстракцією.

Виберіть правильну відповідь:

- Приклад, $x:\alpha\vdash x:\alpha$
- Приклад, $f:(\beta\to\gamma\to\delta),y:\beta\vdash(fy):(\gamma\to\delta)$
- Приклад, $\vdash(\lambda x:\alpha.x):(\alpha\to\alpha)$

Якщо відповідь – третій варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Приклад, $\left|-\left(\lambda x:\alpha.x\right):\left(\alpha\to\alpha\right)\right.$».

Крок 28. Третє правило дозволяє типізувати аплікацію (застосування) за умови, що лівий аплікант має відповідний стрілочний тип.

Виберіть правильну відповідь:

- Приклад, $x:\alpha\left|-x:\alpha\right.$
- Приклад, $f:\left(\beta\to\gamma\to\delta\right),y:\beta\left|-\left(fy\right):\left(\gamma\to\delta\right)\right.$
- Приклад, $\left|-\left(\lambda x:\alpha.x\right):\left(\alpha\to\alpha\right)\right.$

Якщо відповідь – другий варіант, то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку: «Приклад, $f:\left(\beta\to\gamma\to\delta\right),y:\beta\left|-\left(fy\right):\left(\gamma\to\delta\right)\right.$».

Крок 29. Встановіть відповідність. Приклади тверджень про типізації в стилі Черча:

- $x:\alpha\left|-x:\alpha\right.$ (введення $\to$)
- $\left|-\left(\lambda x:\alpha.x\right):\left(\alpha\to\alpha\right)\right.$ (видалення $\to$)
- $f:\left(\beta\to\gamma\to\delta\right),y:\beta\left|-\left(fy\right):\left(\gamma\to\delta\right)\right.$ (аксіома)

Якщо відповідь –

- $x:\alpha\left|-x:\alpha\right.$ (аксіома)
- $\left|-\left(\lambda x:\alpha.x\right):\left(\alpha\to\alpha\right)\right.$ (введення $\to$)
- $f:\left(\beta\to\gamma\to\delta\right),y:\beta\left|-\left(fy\right):\left(\gamma\to\delta\right)\right.$ (видалення $\to$),

то перехід на наступний крок. Якщо інша відповідь – виводиться повідомлення про помилку.

В кінці відображається перелік питань, в якому вказано було вибрано правильну відповідь чи ні. Пропонується закрити тренажер або перейти на стартову сторінку, де можна розпочати тестування спочатку.

3.3. Розробка блок-схеми, яка підлягає програмуванню

На рисунках 3.1 – 3.3 зображена блок-схема алгоритму роботи тренажера.

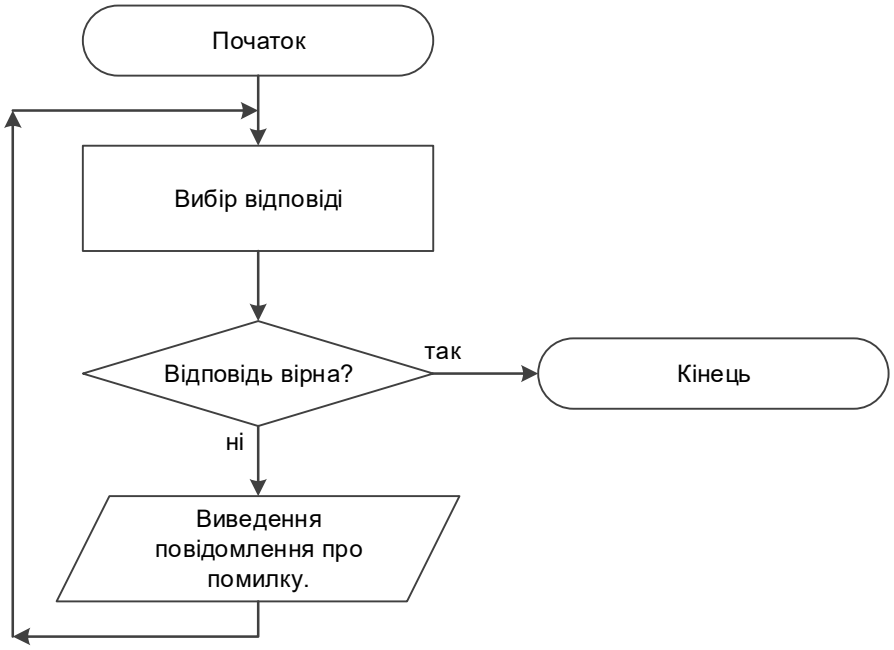

Рисунок 3.1 – Блок-схема процесу «Вибір відповіді»

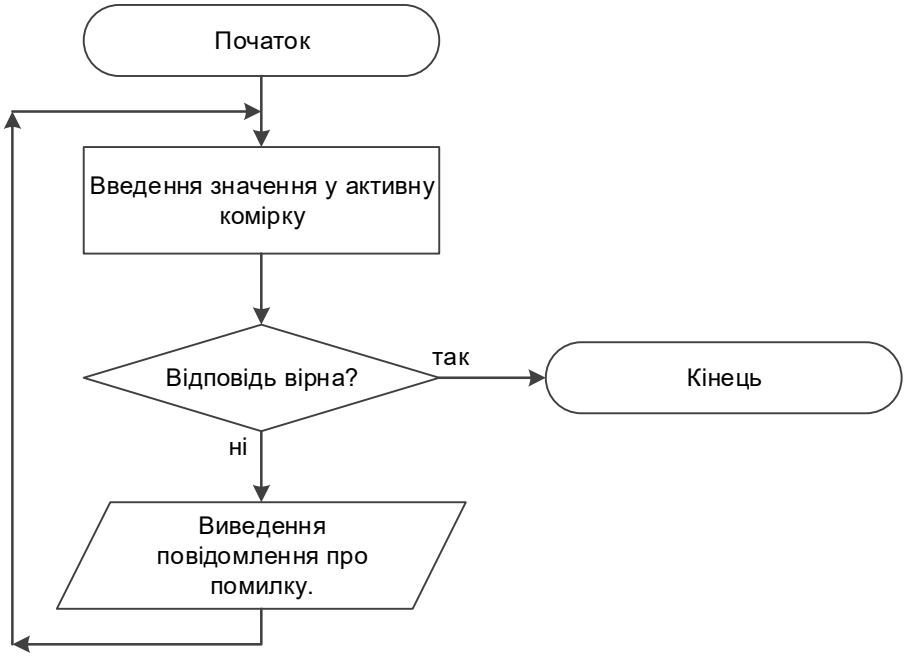

Рисунок 3.2 – Блок-схема процесу «Заповнення комірок»

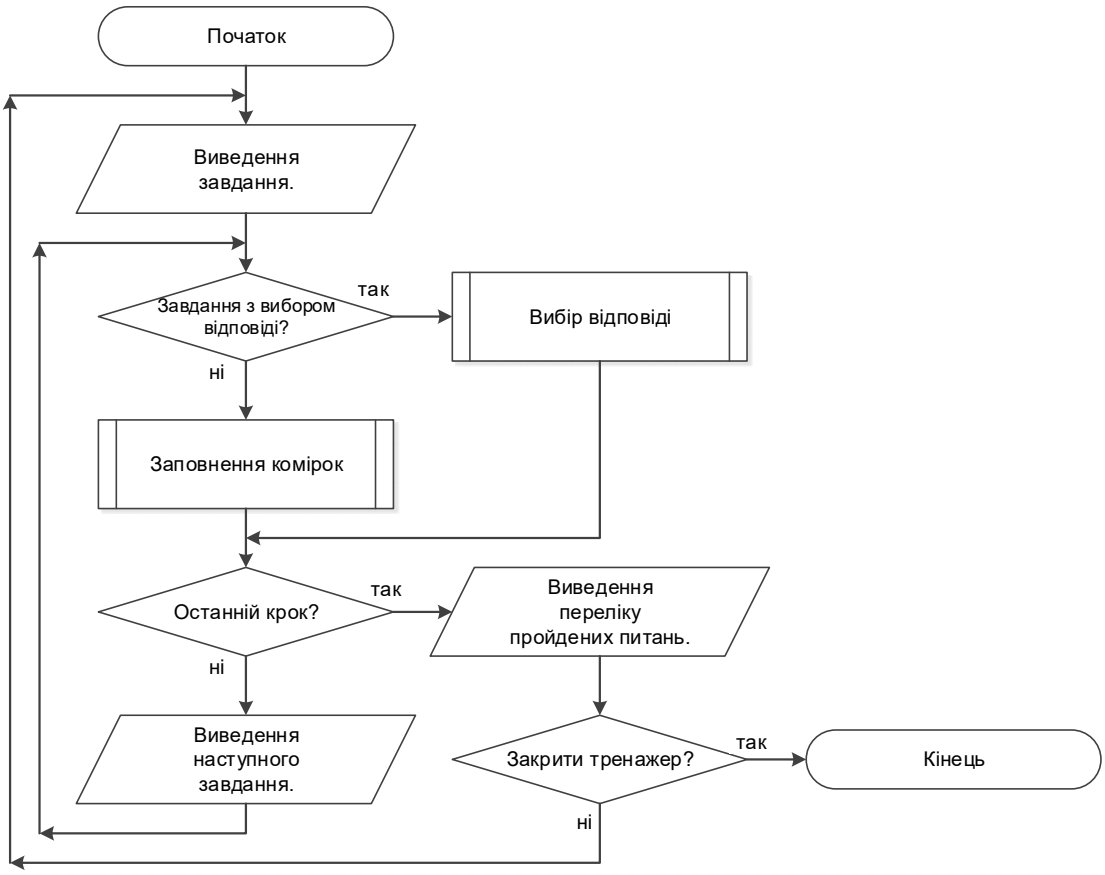

Рисунок 3.3 – Блок-схема алгоритму роботи тренажера

3.4. Обґрунтування вибору програмних засобів для реалізації завдання роботи

Платформа Java - нова програмна платформа для транспортування й виконання високо інтерактивних, динамічних і безпечних аплетів і додатків на системах мережевих комп'ютерів. Основною якістю Java-платформи, що виділяє її серед інших, є те, що вона розташовується на самому верхньому рівні в інших платформах, що дозволяє їй робити компіляцію в байт-коди, не прив'язані до кожної з фізичних машин і представляють собою машинні інструкції для віртуальної машини (virtual machine). Програма, написана мовою Java, компілюється у файл байт-коду, що може працювати скрізь, де присутня Java-Платформа, на кожній з

основних операційних систем. Інакше кажучи, той самий файл буде виконуватися на будь-якій операційній системі, на якій присутня Java-Платформа. Подібна мобільність стає можливою завдяки тому, що в основі Java-платформи лежить віртуальна Java-машина.

У той час як кожна платформа має у своєму розпорядженні свою власну реалізацію віртуальної Java-машини, всі віртуальні машини задовольняють вимогою єдиної специфікації. Завдяки цьому, платформа Java може реалізовувати єдиний стандарт - універсальний програмний інтерфейс для аплетів і додатків на будь-яких апаратних засобах. Тому платформа Java є ідеальною для Інтернету, де та сама програма повинна бути здатна до виконання на будь-якому комп'ютері у світі.

Розробники використовують мову Java при написанні вихідного коду для Java-додатків. Вони компілюють свій код один раз і позбуваються тим самим від необхідності компілювати його для кожної системи окремо. Вихідний текст мови Java компілюється в проміжну, переносну форму байт-коду, що запуститься скрізь, де є присутнім Java-Платформа.

Розробники можуть писати объектно-орієнтовані, багатопоточні, динамічно зв'язані додатки, використовуючи мову Java. Платформа має вбудовані системи захисту, обробки виняткових ситуацій, і автоматичного «збору сміття». Крім того, існує можливість використовувати JIT (just-in-time) компілятори (компілятори «на льоту») і прискорити виконання програм за допомогою перетворення байт-кодів Java у машинну мову. Також, розробники можуть записувати й викликати так звані нативні методи - методи C, C++ або інших мов, відкомпільовані для певної операційної системи - для підвищення швидкості виконання або для застосування спеціальних функціональних можливостей.

Програми, написані мовою Java і потім відкомпільовані, будуть запускатися на Java-Платформі. Платформа Java має дві основних частини:

- Java Virtual Machine (віртуальна Java-Машина);
- Java API (прикладний програмний інтерфейс Java).

У сукупності, ці частини забезпечують оперативні засоби керування роботою програми для кінцевого користувача при установці інтернет-додатків.

The Java Base Platform - «мінімальна» Java платформа, створена для запуску Java-Аплетів і додатків, що розробники можуть без проблем встановити й використовувати. Дана платформа призначається для мережевих, настільних комп'ютерів і робочих станцій. Платформа містить у собі ту ж віртуальну машину, що описувалася вище й при цьому має мінімальний комплект API, необхідним для запуску основних аплетів і додатків. Згаданий мінімальний комплект відомий, як Java Applet API або Java Base API. Розробники, які пишуть для цього комплекту можуть бути впевнені в тім, що програма запуститься скрізь без необхідності в підключенні додаткових бібліотек класів.

Деякі ліцензіати платформи Java уклали контракти про включення приватних реалізацій Java Base API в Java- платформі. В міру розробки бібліотек класів, Java Base Platform розростаються й нові класи регулярно мігрують у встановлену на кожну ліцензійну операційну систему Java Base Platform.

Інший набір API, що називається Standard Extension API, визначений JavaSoft у партнерстві з провідними промисловими компаніями створений для розширення основних функціональних можливостей.

Embedded Java Platform була розроблена для споживача, що використовує прилади з малими ресурсами й з більш спеціалізованою функціональністю, ніж мережевий комп'ютер. Наприклад, принтери, ксерокси, мобільні телефони й ін. Подібна апаратура може мати деякі специфічні властивості, а саме невеликий обсяг пам'яті, відсутність дисплея або неможливість зв'язку по мережі.

API, розроблений для такої платформи, називається Java Embedded API. Java Embedded API - найменший із прикладних програмних

інтерфейсів, які можуть бути впроваджені в описані вище прилади й при цьому ефективно працювати. Оскільки дана платформа усе ще допрацьовується, Java-додатки, написані для одного окремого пристрою, зберігають працездатність на широкому діапазоні подібних по своїй специфіці пристроїв [11].

4. ПРАКТИЧНА ЧАСТИНА

4.1. Опис процесу програмної реалізації

Після створення проекту потрібно додати форму JFrame, що використовується для відображення програми. Розмір форми не обов’язково відразу встановлювати, його можна змінювати в процесі розробки тренажера (рис. 4.1).

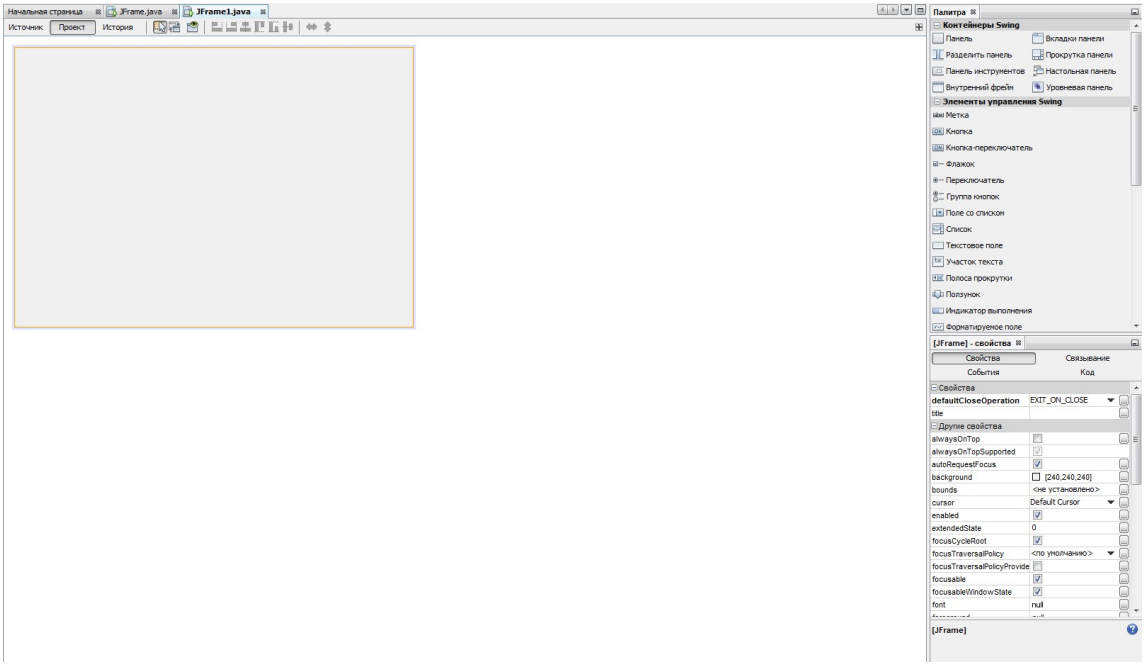

Рисунок 4.1 – Створено новий проект

Тепер на ньому розміщуються інші елементи, що можна додати через контекстне меню (рис. 4.2) або з панелі інструментів.

Першим чином потрібно додати контейнер JPanel і встановити йому карткове розміщення (рис. 4.3) для можливості виводити один з контейнерів: стартова сторінка, сторінка із завданням, кінцева сторінка. Також так реалізується відображення правильної форми при виведенні завдань.

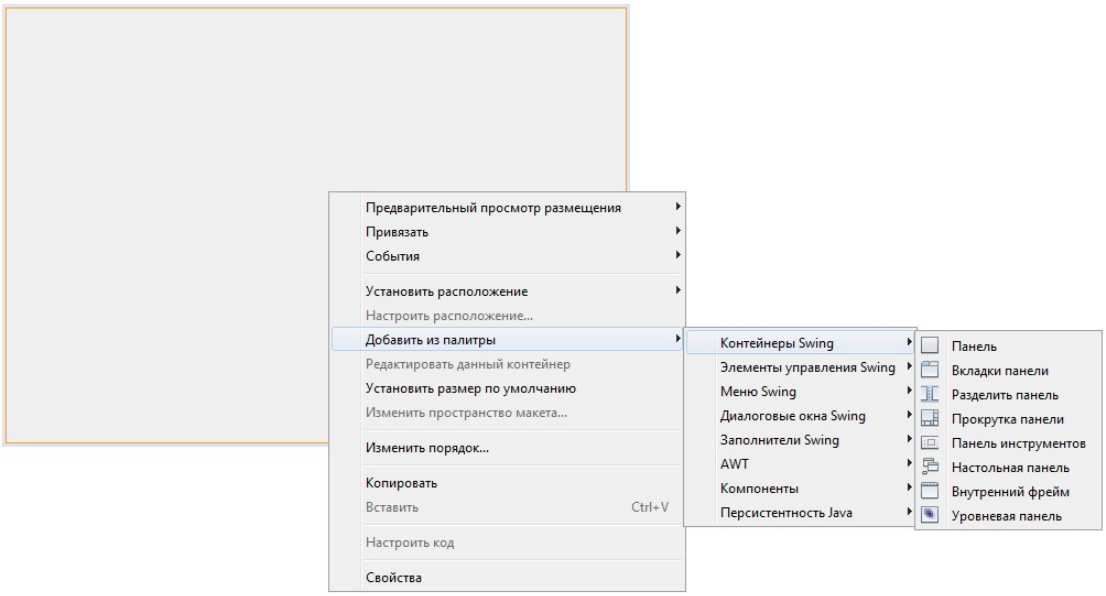

Рисунок 4.2 – Контекстне меню

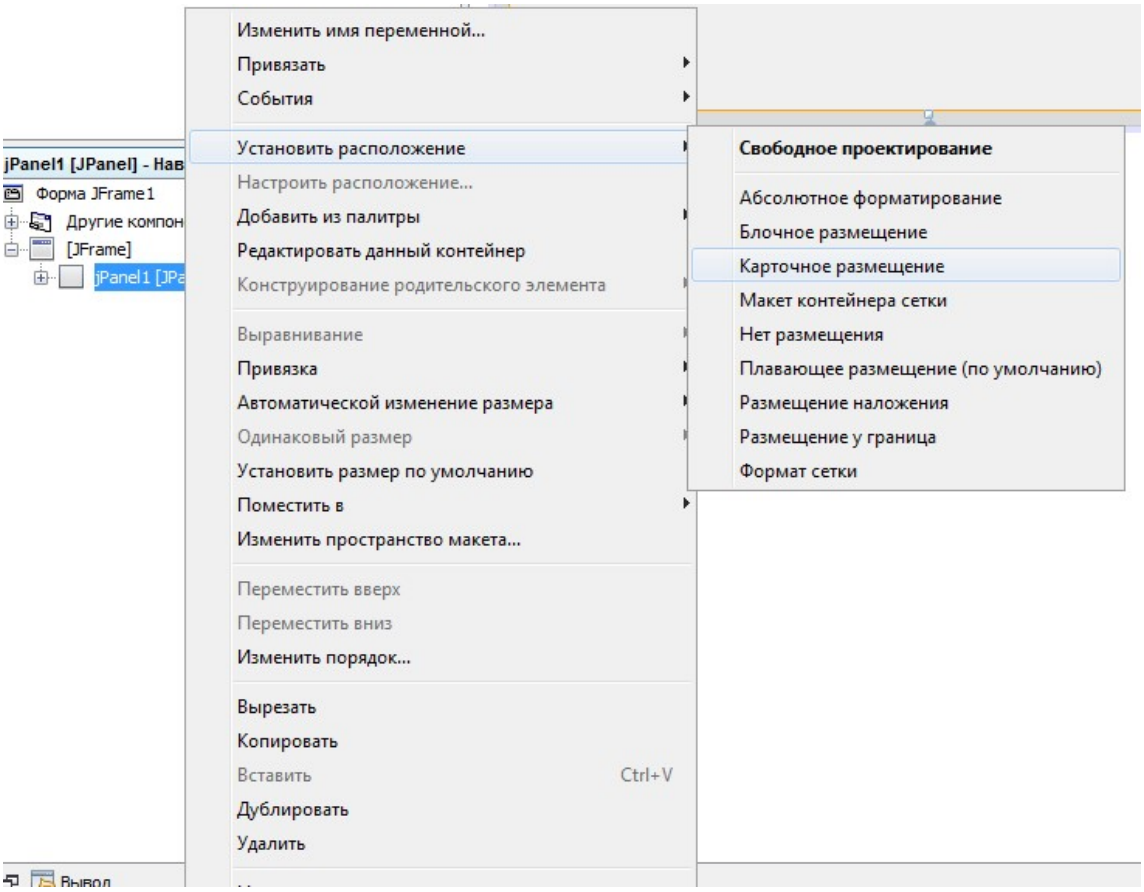

Рисунок 4.3 – Встановлення карткового розміщення

Після цього вже було розміщено інші елементи.

Наступним кроком був етап програмування. Для створення і відображення форми:

```
    public JFrame() {
        initComponents();
    }

    public static void main(String args[]) {
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new JFrame().setVisible(true);
            }
        });
    }
```

Серед бібліотек було підключено наступні:

```
import java.awt.CardLayout;
import java.awt.Desktop;
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
```

Оскільки в тренажері передбачено можливість завантаження файлу, то для цього було створено функцію String DownloadFile(). При цьому використовуються BufferedInputStream і FileOutputStream, а як результат виводиться шлях до файлу.

```
    private String DownloadFile() {
        String url = System.getProperty("user.dir");
        BufferedInputStream in = null;
        FileOutputStream fout = null;
        try {
            in = new
BufferedInputStream(getClass().getResource("Theory.pdf").openStream());
            fout = new FileOutputStream("Theory.pdf");
            byte data[] = new byte[1024];
            int count;
            while ((count = in.read(data, 0, 1024)) != -1) {
                fout.write(data, 0, count);
            }
        } catch (IOException ex) {
            Logger.getLogger(JFrame.class.getName()).log(Level.SEVERE,
null, ex);
        } finally {
            try {
                if ( in != null)
```

```
                    in.close();
                if (fout != null) {
                    fout.close();
                }
            }
            catch (IOException ex) {
                Logger.getLogger(JFrame.class.getName()).log(Level.SEVERE,
null, ex);
            }
        }
        url+="\\Theory.pdf";

        return url;
    }
```

Для відображення потрібного контейнера реалізовано void ChangeWindow(JPanel p, String s), де JPanel p – батьківський контейнер, String s – назва картки дочірнього контейнера.

```
    private void ChangeWindow(JPanel p, String s) {
        CardLayout cl =(CardLayout) p.getLayout();
        cl.show(p, s);
    }
```

Виведення повідомлення про помилку виконується функцією void ShowError(String s) завдяки JOptionPane.showMessageDialog. В параметрі вказується ім’я зображення.

```
    private void ShowError(String s) {
        ImageIcon image = new
ImageIcon(getClass().getResource("/images/"+s+".png"));
        JLabel label = new JLabel();
        label.setIcon(image);
        JOptionPane.showMessageDialog(jPanel1,label,"Допущено
помилку!",JOptionPane.PLAIN_MESSAGE);
    }
```

Останні дві функції створено для відображення завдання. Перша – void ChooseTask(String s1,String s2,String s3,String s4), що змінює вміст завдань з вибором відповіді. В параметрах вказуються імена зображень. Якщо String s4 вказати пустим, то третій варіант відповіді не виводиться.

```
    private void ChooseTask(String s1,String s2,String s3,String s4) {
        buttonGroup1.clearSelection();
        jLabel2.setIcon(new
ImageIcon(getClass().getResource("/images/"+s1+".png")));
        jLabel4.setIcon(new
ImageIcon(getClass().getResource("/images/"+s2+".png")));
        jLabel5.setIcon(new
ImageIcon(getClass().getResource("/images/"+s3+".png")));
        if(s4.isEmpty()) {
            jRadioButton3.setVisible(false);
```

```
            jLabel6.setVisible(false);
        } else {
            if(!jRadioButton3.isVisible()) {
                jRadioButton3.setVisible(true);
                jLabel6.setVisible(true);
            }
            jLabel6.setIcon(new
ImageIcon(getClass().getResource("/images/"+s4+".png")));
        }

    }
```

Друга – void WriteTask(String s1,String s2,boolean b) змінює вміст завдань з введенням відповіді. Якщо boolean b вказано true, то виводиться два поля для відповіді. Якщо false, то одне поле.

```
    private void WriteTask(String s1,String s2,boolean b) {
        jLabel7.setIcon(new
ImageIcon(getClass().getResource("/images/"+s1+".png")));
        jLabel9.setIcon(new
ImageIcon(getClass().getResource("/images/"+s2+".png")));
        jTextField1.setText("");
        jTextField2.setText("");
        if(b) {
            jTextField2.setVisible(true);
        } else {
            jTextField2.setVisible(false);
        }
    }
```

Було створено дві змінні: int step – поточний крок; boolean[] steps – масив булевих значень для позначення, на яких кроках було допущено помилку.

```
    int step;
    boolean[] steps = new boolean[29];
```

Залишилось реалізувати події при натисненні на кнопки.

Щоб відкрити вікно з теоретичним матеріалом застосовується theoryWActionPerformed. При цьому обов’язково потрібно вказати розмір ділового вікна.

```
    private void theoryWActionPerformed(java.awt.event.ActionEvent evt) {
        JDialog d = jDialog1;
        d.setSize(655,700);
        d.setResizable(false);
        d.setVisible(true);
    }
```

Подія theoryFActionPerformed відповідає за збереження теоретичного матеріалу як файлу. В даному випадку використовується функція DownloadFile().

```
private void theoryFActionPerformed(java.awt.event.ActionEvent evt) {
    String res = DownloadFile();
    jLabel16.setText(res);
}
```

Для переходу від початкової сторінки до виконання тренажера застосовується startBActionPerformed. Змінній step присвоюється значення 1, використовуються функції ChooseTask(String s1,String s2,String s3,String s4) і ChangeWindow(JPanel p, String s).

```
private void startBActionPerformed(java.awt.event.ActionEvent evt) {
    step = 1;
    ChooseTask("s1", "s1-1", "s1-2", "s1-3");
    ChangeWindow(jPanel1, "task");
    ChangeWindow(jPanel2, "step1");
}
```

Щоб працювали перевірка відповіді і перехід до наступного кроку застосовується nextBActionPerformed. Завдання з вибором відповіді перевіряються завдяки jRadioButton.isSelected(), а з введенням – equals(jTextField1.getText()) і equals(jTextField2.getText()). При помилці змінюється відповідне значення масиву steps[] на true (див. Додаток А).

```
private void nextBActionPerformed(java.awt.event.ActionEvent evt) {
    switch (step) {
        case 1:
            if(jRadioButton1.isSelected()) {
                ChooseTask("s2", "s2-1", "s2-2", "s2-3");
                step++;
            } else {
                ShowError("s1-er");
                steps[step-1] = true;
            }
```

Тренажер закривається подією closeBActionPerformed.

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

Перехід до стартової сторінки здійснюється завдяки homeBActionPerformed.

```
private void homeBActionPerformed(java.awt.event.ActionEvent evt) {
    ChangeWindow(jPanel1, "start");
}
```

4.2. Опис програми

На стартовій сторінці виводиться основна інформація про тренажер, можливість відкрити теоретичний матеріал у вікні чи зберегти його (рис. 4.4).

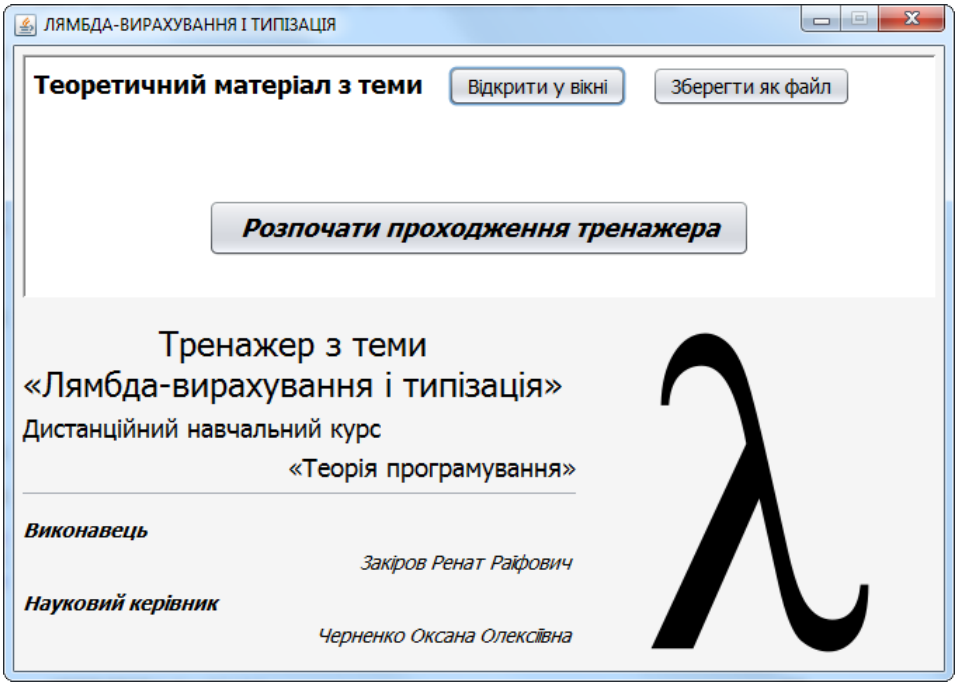

Рисунок 4.4 – Стартова сторінка

Якщо відкрити матеріал у вікні, то відобразиться діалогове вікно (рис. 4.5). Якщо зберегти як файл, то виведеться шлях до нього (рис. 4.6).

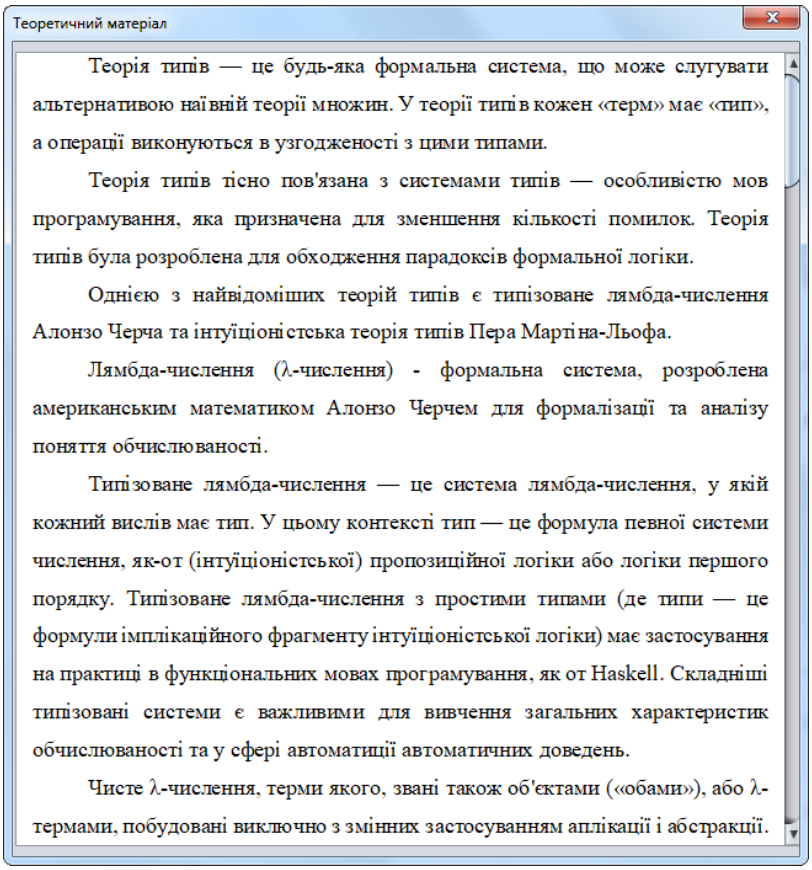

Рисунок 4.5 – Виведення теоретичного матеріалу

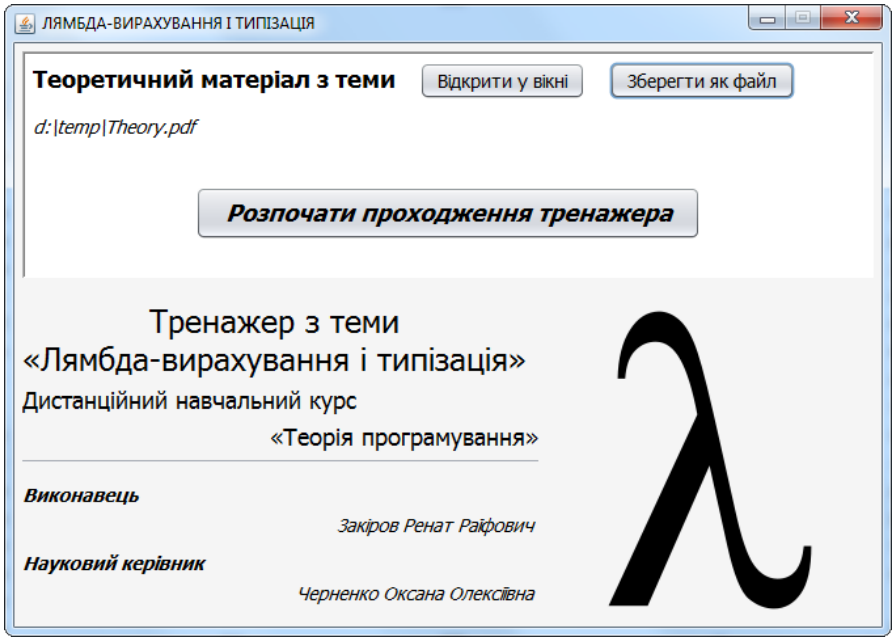

Рисунок 4.6 – Збереження теоретичного матеріалу

Після початку виконання тренажера виведеться перше завдання, в якому потрібно вибрати відповідь (рис. 4.7). В такому виді завдань надається вибір серед двох або трьох варіантів відповіді.

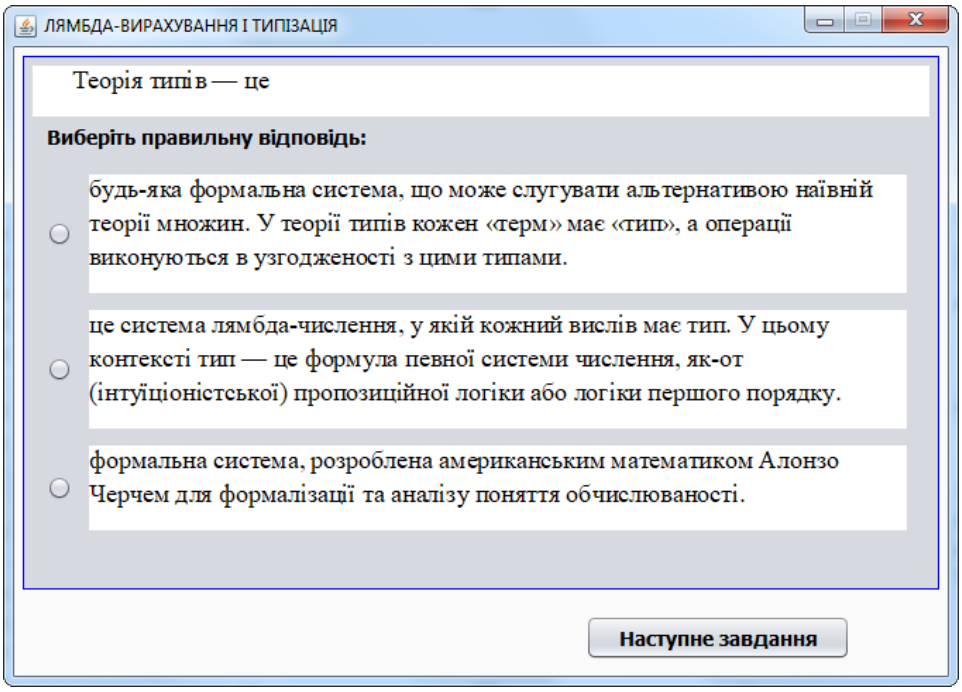

Рисунок 4.7 – Вибір відповіді

Якщо було допущено помилку, то з’явиться повідомлення про неї (рис. 4.8). При правильній відповіді відбудеться перехід до наступного завдання.

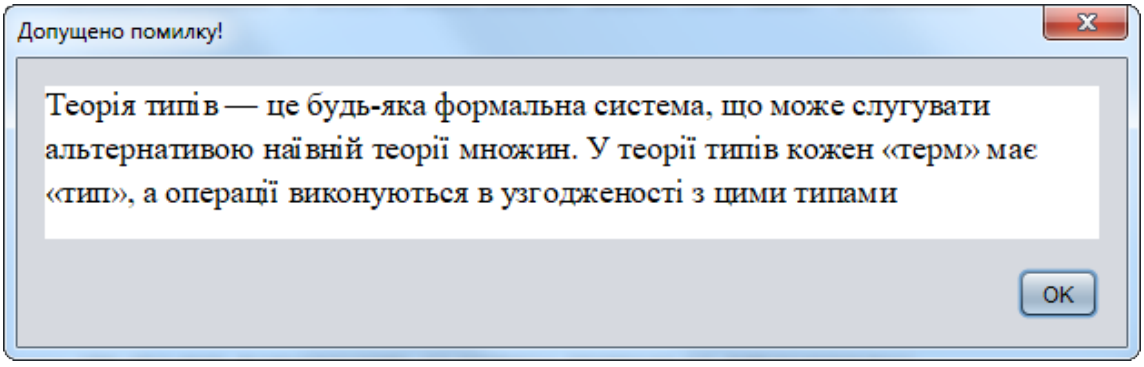

Рисунок 4.8 – Повідомлення про помилку

Інший вид завдань – це введення відповіді. Аналогічно попередньому в ньому може бути одне або два поля для заповнення (рис. 4.9).

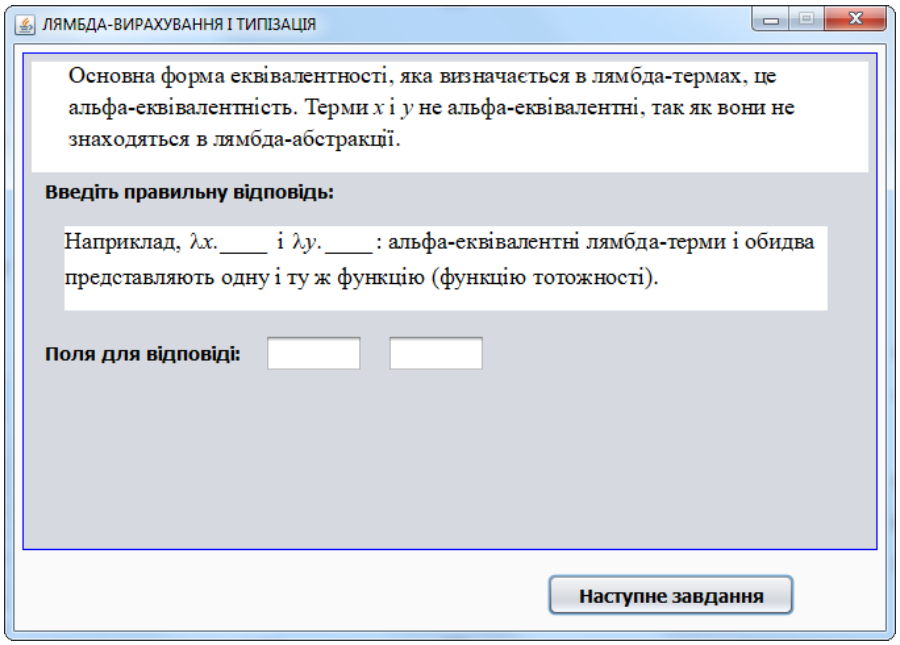

Рисунок 4.9 – Введення відповіді

Останній вид – встановлення відповідності (рис. 4.10).

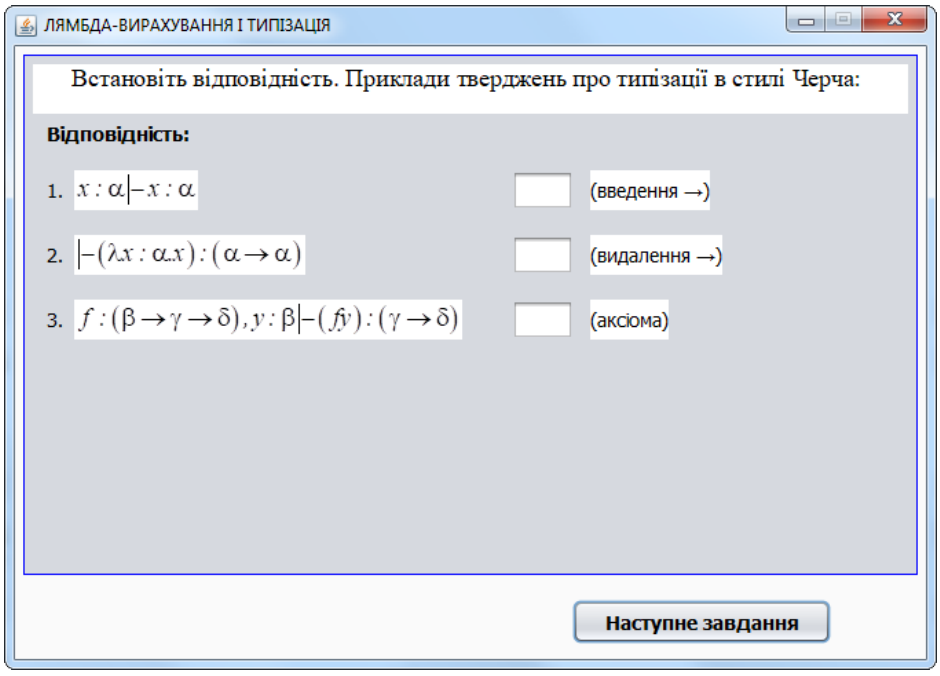

Рисунок 4.10 – Встановлення відповідності

На кінцевій сторінці відображається перелік питань, в якому вказано було вибрано правильну відповідь чи ні (рис. 4.11).

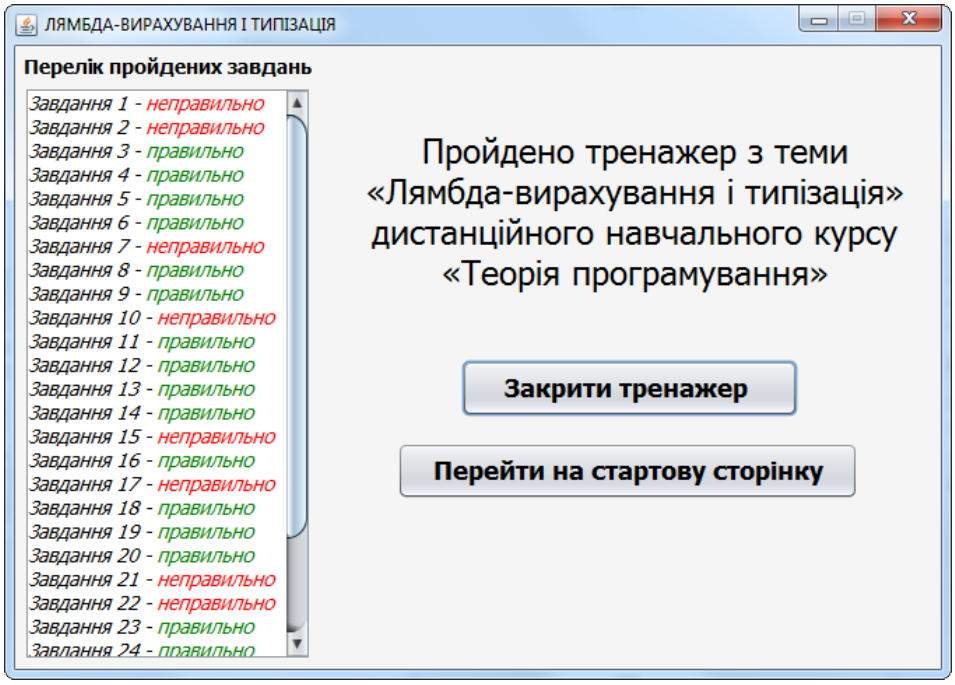

Рисунок 4.11 – Кінцева сторінка

4.3. Необхідна користувачу програми інструкція

Щоб тренажер запустився потрібно щоб на комп'ютері була встановлена Java. Її можна завантажити з офіційного сайту https://java.com/ru/download/.

На стартовій сторінці тренажера можна переглянути теоретичний матеріал. Для цього потрібно натиснути кнопку «Відкрити у вікні». Якщо ж бажаєте його зберегти, то натиснути «Зберегти як файл» (рис. 4.5, 4.6).

Для проходження тренажера необхідно натиснути «Розпочати проходження тренажера». Відображається перше завдання. Потрібно вибрати один з варіантів і натиснути кнопку «Наступне завдання» (рис. 4.12).

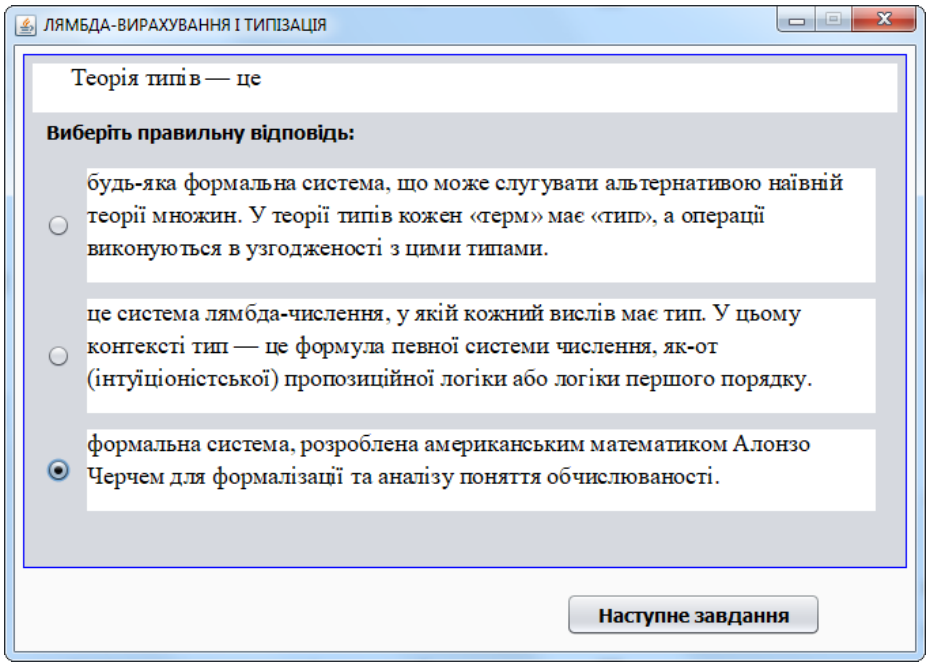

Рисунок 4.12 – Перше завдання

При неправильній відповіді відображається повідомлення про помилку, що вказує на вірну відповідь (рис. 4.8). Якщо правильна, то перехід до наступного завдання (рис. 4.13).

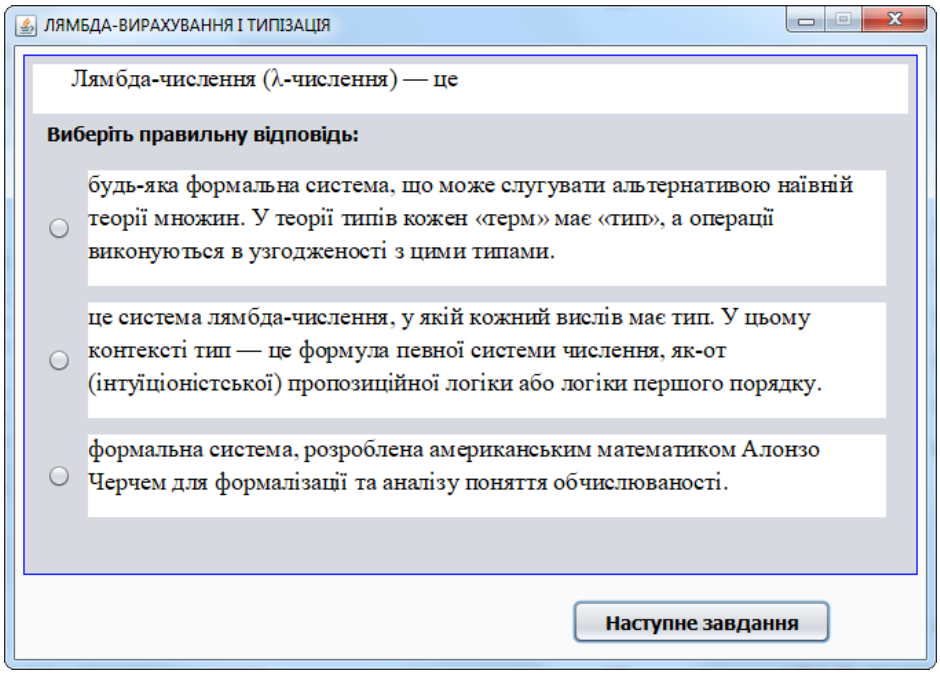

Рисунок 4.13 – Друге завдання

Також можуть з’являтися завдання, в яких потрібно вводити відповідь (рис. 4.9). В такому випадку заповнюються всі доступні поля (рис. 4.14).

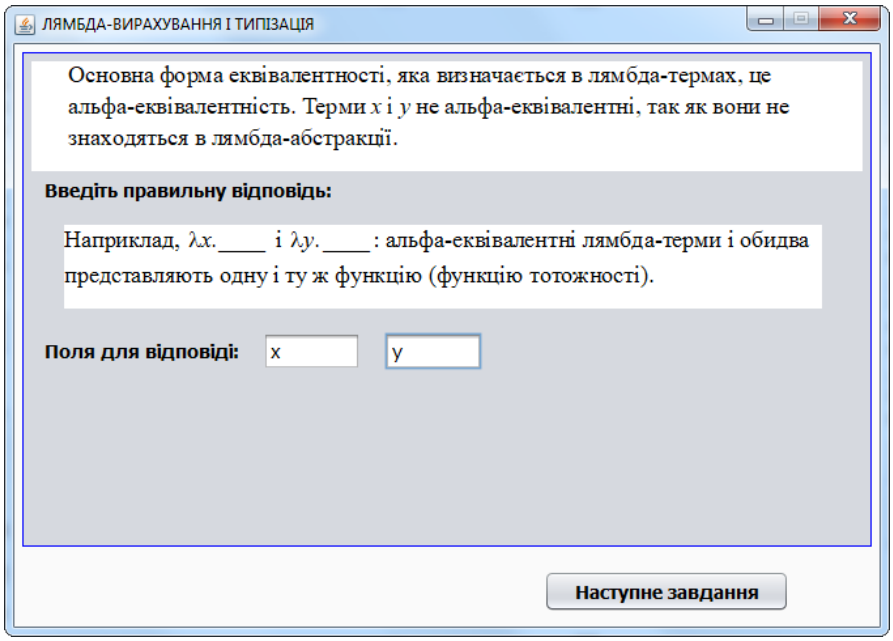

Рисунок 4.14 – Шосте завдання

В завданні на встановлення відповідності необхідно вказати в полях відповідні номера формул (рис. 4.15).

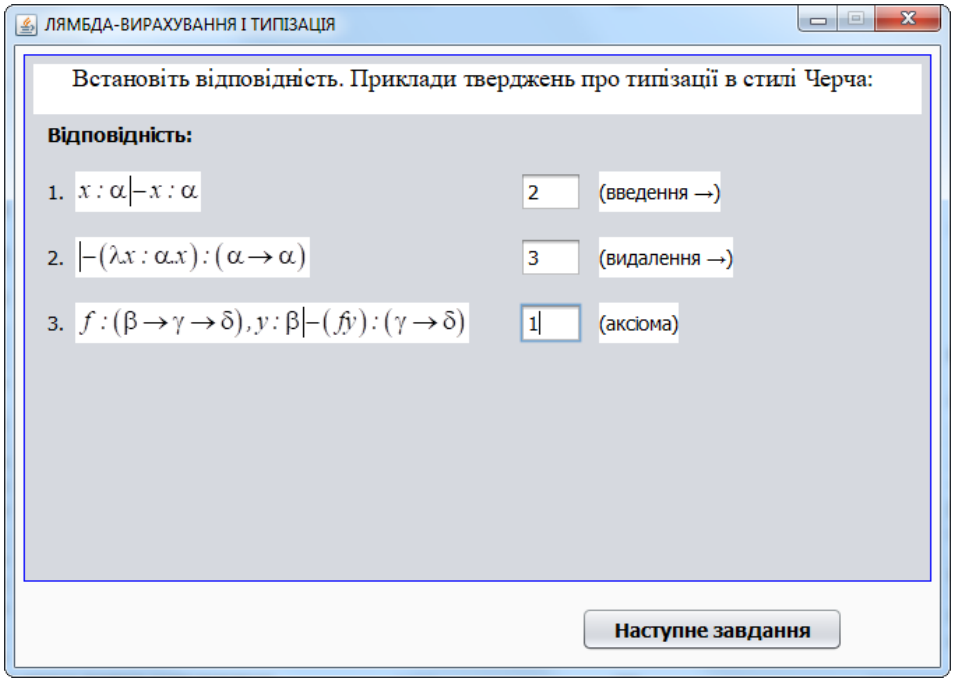

Рисунок 4.15 – Останнє завдання

В кінці відображається перелік питань, в якому вказано було вибрано правильну відповідь чи ні (рис. 4.11). Його можна листати за допомогою панелі прокрутки. Всі пункти в списку позначаються відповідним кольором.

Пропонується закрити тренажер. Для цього слід натиснути кнопку «Закрити тренажер». Щоб перейти на стартову сторінку, де можна розпочати тестування спочатку, потрібно натиснути «Перейти на стартову сторінку».

ВИСНОВКИ

Метою виконаної роботи є розроблений тренажер з теми «Лямбда-вирахування і типізація» дистанційного навчального курсу «Теорія програмування».

Розглянемо основні завдання роботи:

- описано основні вимоги до тренажера;
- розглянуто актуальність використання дистанційного навчання;
- наведено теоретичний матеріал з теми для його використання в тренажері;
- розроблено алгоритм тренажера;
- розроблено блок-схему розробленого алгоритму;
- описано причини вибору мови програмування для реалізації тренажера;
- розроблено тренажер з даної теми;
- описано процес реалізації тренажера;
- описано роботу тренажера

Згідно алгоритму на стартовій сторінці пропонується переглянути теоретичний матеріал за темою або завантажити його. Також виводиться наступна інформація:

- тема тренажера;
- назва дисципліни;
- прізвище, ініціали розробника;
- прізвище, ініціали керівника.

На кожному кроці виводиться завдання та варіанти відповіді, серед яких слід вибрати одну правильну. При неправильній відповіді відображається повідомлення про помилку, що вказує на вірну відповідь.

На останньому кроці відображається перелік питань, в якому вказано було вибрано правильну відповідь чи ні. Пропонується закрити тренажер

або перейти на стартову сторінку, де можна розпочати тестування спочатку.

Безумовно, як і в кожній формі отримання знань, у дистанційній є і свої недоліки, але їх подолання стає можливим завдяки рокам практичного застосування цієї форми не лише як допоміжної та однієї з побічних, а як можливо рівної класичній формі здобуття освіти.

Зважаючи на викладене вище, ми можемо спрогнозувати певні тенденції розвитку дистанційного навчання, такі як збільшення кількості масових відкритих дистанційних курсів, розробка програм дистанційного навчання, інтеграція інтелектуальних комп’ютерних технологій у навчальний процес дистанційної освіти, комбінування переваг дистанційного навчання із класичною формою освіти, моніторинг досягнень вищих навчальних закладів не лише в межах України, а і в усьому світі і подальше використання корисного досвіду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ємець О. О. Методичні рекомендації до виконання бакалаврської роботи для студентів за 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, 2011. – 71 с.
2. Дистанційне навчання у вищих навчальних закладах України : [Інформаційні матеріали] / МОН України. – Хмельницький : ХНУ, 2009. – 50 с.
3. Про Основні засади розвитку інформаційного суспільства в Україні на 2007-2015 роки : Закон України від 9.01.2007 № 537-V [Електронний ресурс]. – Режим доступу : http://zakon.rada.gov.ua/cgi-bin/laws/main.cgi?nreg=537-16.
4. Офіційні матеріали наради-семінару з питань нормативного забезпечення дистанційної форми навчання в Україні ; Національний технічний університет України «КПІ», м. Київ, 2012 [Електронний ресурс]. – Режим доступу : http://ipo.kpi.ua/ua/distance/dlabout.html
5. Комп'ютерні науки і прикладна математика (КНіПМ-2018): матеріали науково- практичного семінару. Випуск 1 / за ред. Ємця О.О. – Полтава: Кафедра ММСІ ПУЕТ, 2018. – 64 с.
6. Інформатика та системні науки (ІСН – 2017) : матеріали VIII Все української науково-практичної конференції за міжнародною участю (м. Полтава, 16–18 березня 2017 р.) / за ред. Ємця О. О. – Полтава : ПУЕТ, 2017. – 333 с.
7. Черненко О.О. Електронний навчально-методичний посібник для самостійного вивчення навчальної дисципліни «Теорія програмування» для студентів напряму 6.040302 «Інформатика».
8. Бабій М.С. Теорія програмування: Навчальний посібник [Електронний ресурс] / М.С. Бабій, О.П. Чекалов.– Суми: Вид-во СумДУ, 2009. – 181 с.

9. Лямбда-исчисление / Материал из Википедии — свободной энциклопедии [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Лямбда-исчисление
10. Просто типизированное лямбда-исчисление / Материал из Википедии — свободной энциклопедии [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/Просто_типизированное_лямбда-исчисление
11. Монахов Вадим Язык программирования Java и среда NetBeans. / Вадим Монахов— 3-е изд. — СПб.: БХВ-Петербург, 2011. — 704 с.
12. 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

```
package lambda;

import java.awt.CardLayout;
import java.awt.Desktop;
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.ImageIcon;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

public class JFrame extends javax.swing.JFrame {

   /**
    * Creates new form JFrame
    */
   public JFrame() {
      initComponents();
   }

   private String DownloadFile() {
      String url = System.getProperty("user.dir");
      BufferedInputStream in = null;
      FileOutputStream fout = null;
      try {
         in = new BufferedInputStream(getClass().getResource("Theory.pdf").openStream());
         fout = new FileOutputStream("Theory.pdf");
         byte data[] = new byte[1024];
         int count;
         while ((count = in.read(data, 0, 1024)) != -1) {
            fout.write(data, 0, count);
         }
      } catch (IOException ex) {
         Logger.getLogger(JFrame.class.getName()).log(Level.SEVERE, null, ex);
      } finally {
         try {
            if ( in != null)
               in.close();
            if (fout != null) {
               fout.close();
            }
```

```
            }
            catch (IOException ex) {
                Logger.getLogger(JFrame.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        url+="\\Theory.pdf";

        return url;
    }

    private void ChangeWindow(JPanel p, String s) {
        CardLayout cl =(CardLayout) p.getLayout();
        cl.show(p, s);
    }

    private void ShowError(String s) {
        ImageIcon image = new ImageIcon(getClass().getResource("/images/"+s+".png"));
        JLabel label = new JLabel();
        label.setIcon(image);
        JOptionPane.showMessageDialog(jPanel1,label,"Допущено
помилку!",JOptionPane.PLAIN_MESSAGE);
    }

    private void ChooseTask(String s1,String s2,String s3,String s4) {
        buttonGroup1.clearSelection();
        jLabel2.setIcon(new ImageIcon(getClass().getResource("/images/"+s1+".png")));
        jLabel4.setIcon(new ImageIcon(getClass().getResource("/images/"+s2+".png")));
        jLabel5.setIcon(new ImageIcon(getClass().getResource("/images/"+s3+".png")));
        if(s4.isEmpty()) {
            jRadioButton3.setVisible(false);
            jLabel6.setVisible(false);
        } else {
            if(!jRadioButton3.isVisible()) {
                jRadioButton3.setVisible(true);
                jLabel6.setVisible(true);
            }
            jLabel6.setIcon(new ImageIcon(getClass().getResource("/images/"+s4+".png")));
        }

    }

    private void WriteTask(String s1,String s2,boolean b) {
        jLabel7.setIcon(new ImageIcon(getClass().getResource("/images/"+s1+".png")));
        jLabel9.setIcon(new ImageIcon(getClass().getResource("/images/"+s2+".png")));
        jTextField1.setText("");
        jTextField2.setText("");
        if(b) {
            jTextField2.setVisible(true);
        } else {
```

```
        jTextField2.setVisible(false);
    }
}

int step;
boolean[] steps = new boolean[29];

        private void startBActionPerformed(java.awt.event.ActionEvent evt) {
    step = 1;
    ChooseTask("s1", "s1-1", "s1-2", "s1-3");
    ChangeWindow(jPanel1, "task");
    ChangeWindow(jPanel2, "step1");
}

private void theoryWActionPerformed(java.awt.event.ActionEvent evt) {
    JDialog d = jDialog1;
    d.setSize(655,700);
    d.setResizable(false);
    d.setVisible(true);
}

private void theoryFActionPerformed(java.awt.event.ActionEvent evt) {
    String res = DownloadFile();
    jLabel16.setText(res);
}

private void nextBActionPerformed(java.awt.event.ActionEvent evt) {
    switch (step) {
        case 1:
            if(jRadioButton1.isSelected()) {
                ChooseTask("s2", "s2-1", "s2-2", "s2-3");
                step++;
            } else {
                ShowError("s1-er");
                steps[step-1] = true;
            }
            break;
        case 2:
            if(jRadioButton3.isSelected()) {
                ChooseTask("s3", "s3-1", "s3-2", "s3-3");
                step++;
            } else {
                ShowError("s2-er");
                steps[step-1] = true;
            }
            break;
        case 3:
            if(jRadioButton2.isSelected()) {
                ChooseTask("s4", "s4-1", "s4-2", "s4-3");
```

```
            step++;
        } else {
            ShowError("s3-er");
            steps[step-1] = true;
        }
        break;
    case 4:
        if(jRadioButton2.isSelected()) {
            ChooseTask("s5", "s5-1", "s5-2", "s5-3");
            step++;
        } else {
            ShowError("s4-er");
            steps[step-1] = true;
        }
        break;
    case 5:
        if(jRadioButton1.isSelected()) {
            WriteTask("s6", "s6-1",true);
            ChangeWindow(jPanel2, "step2");
            step++;
        } else {
            ShowError("s5-er");
            steps[step-1] = true;
        }
        break;
    case 6:
        if("x".equals(jTextField1.getText()) && "y".equals(jTextField2.getText())) {
            WriteTask("s7", "s7-1",false);
            step++;
        } else {
            ShowError("s6-er");
            steps[step-1] = true;
        }
        break;
    case 7:
        if("t[x:=a]".equals(jTextField1.getText())) {
            WriteTask("s8", "s8-1",false);
            step++;
        } else {
            ShowError("s7-er");
            steps[step-1] = true;
        }
        break;
    case 8:
        if("f".equals(jTextField1.getText())) {
            WriteTask("s9", "s9-1",false);
            step++;
        } else {
            ShowError("s8-er");
```

```
            steps[step-1] = true;
        }
        break;
    case 9:
        if("x+y".equals(jTextField1.getText())) {
            ChooseTask("s10", "s10-1", "s10-2", "s10-3");
            ChangeWindow(jPanel2, "step1");
            step++;
        } else {
            ShowError("s9-er");
            steps[step-1] = true;
        }
        break;
    case 10:
        if(jRadioButton1.isSelected()) {
            ChooseTask("s11", "s11-1", "s11-2", "s11-3");
            step++;
        } else {
            ShowError("s10-er");
            steps[step-1] = true;
        }
        break;
    case 11:
        if(jRadioButton3.isSelected()) {
            ChooseTask("s12", "s12-1", "s12-2", "");
            step++;
        } else {
            ShowError("s11-er");
            steps[step-1] = true;
        }
        break;
    case 12:
        if(jRadioButton2.isSelected()) {
            ChooseTask("s13", "s13-1", "s13-2", "");
            step++;
        } else {
            ShowError("s12-er");
            steps[step-1] = true;
        }
        break;
    case 13:
        if(jRadioButton1.isSelected()) {
            ChooseTask("s14", "s14-1", "s14-2", "s14-3");
            step++;
        } else {
            ShowError("s13-er");
            steps[step-1] = true;
        }
        break;
```

```
case 14:
   if(jRadioButton3.isSelected()) {
      ChooseTask("s15", "s15-1", "s15-2", "s15-3");
      step++;
   } else {
      ShowError("s14-er");
      steps[step-1] = true;
   }
   break;
case 15:
   if(jRadioButton2.isSelected()) {
      ChooseTask("s16", "s16-1", "s16-2", "s16-3");
      step++;
   } else {
      ShowError("s15-er");
      steps[step-1] = true;
   }
   break;
case 16:
   if(jRadioButton1.isSelected()) {
      ChooseTask("s17", "s17-1", "s17-2", "s17-3");
      step++;
   } else {
      ShowError("s16-er");
      steps[step-1] = true;
   }
   break;
case 17:
   if(jRadioButton3.isSelected()) {
      ChooseTask("s18", "s18-1", "s18-2", "s18-3");
      step++;
   } else {
      ShowError("s17-er");
      steps[step-1] = true;
   }
   break;
case 18:
   if(jRadioButton2.isSelected()) {
      ChooseTask("s19", "s19-1", "s19-2", "s19-3");
      step++;
   } else {
      ShowError("s18-er");
      steps[step-1] = true;
   }
   break;
case 19:
   if(jRadioButton1.isSelected()) {
      ChooseTask("s20", "s20-1", "s20-2", "s20-3");
      step++;
```

```
    } else {
      ShowError("s19-er");
      steps[step-1] = true;
    }
    break;
  case 20:
    if(jRadioButton2.isSelected()) {
      ChooseTask("s21", "s21-1", "s21-2", "s21-3");
      step++;
    } else {
      ShowError("s20-er");
      steps[step-1] = true;
    }
    break;
  case 21:
    if(jRadioButton1.isSelected()) {
      ChooseTask("s22", "s22-1", "s22-2", "s22-3");
      step++;
    } else {
      ShowError("s21-er");
      steps[step-1] = true;
    }
    break;
  case 22:
    if(jRadioButton1.isSelected()) {
      ChooseTask("s23", "s23-1", "s23-2", "s23-3");
      step++;
    } else {
      ShowError("s22-er");
      steps[step-1] = true;
    }
    break;
  case 23:
    if(jRadioButton2.isSelected()) {
      ChooseTask("s24", "s24-1", "s24-2", "s24-3");
      step++;
    } else {
      ShowError("s23-er");
      steps[step-1] = true;
    }
    break;
  case 24:
    if(jRadioButton3.isSelected()) {
      ChooseTask("s25", "s25-1", "s25-2", "s25-3");
      step++;
    } else {
      ShowError("s24-er");
      steps[step-1] = true;
    }
```

```
            break;
        case 25:
            if(jRadioButton3.isSelected()) {
                ChooseTask("s26", "s26-1", "s26-2", "s26-3");
                step++;
            } else {
                ShowError("s25-er");
                steps[step-1] = true;
            }
            break;
        case 26:
            if(jRadioButton1.isSelected()) {
                ChooseTask("s27", "s26-1", "s26-2", "s26-3");
                step++;
            } else {
                ShowError("s26-er");
                steps[step-1] = true;
            }
            break;
        case 27:
            if(jRadioButton3.isSelected()) {
                ChooseTask("s28", "s26-1", "s26-2", "s26-3");
                step++;
            } else {
                ShowError("s27-er");
                steps[step-1] = true;
            }
            break;
        case 28:
            if(jRadioButton2.isSelected()) {
                jTextField3.setText("");
                jTextField4.setText("");
                jTextField5.setText("");
                ChangeWindow(jPanel2, "step3");
                step++;
            } else {
                ShowError("s28-er");
                steps[step-1] = true;
            }
            break;
        case 29:
            if("2".equals(jTextField3.getText()) && "3".equals(jTextField4.getText())
                    && "1".equals(jTextField5.getText())) {
                String var="<html>";
                for (int i=0; i<steps.length; i++) {
                    var+="Завдання "+(i+1)+" - <font
color=\""+(steps[i]?"red":"green")+"\">"+(steps[i]?"неправильно":"правильно")+"</font><
br>";
                }
```

```
            jLabel12.setText(var);
            ChangeWindow(jPanel1, "end");
         } else {
            ShowError("s29-er");
            steps[step-1] = true;
         }
         break;
   }
}

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

private void homeBActionPerformed(java.awt.event.ActionEvent evt) {
   ChangeWindow(jPanel1, "start");
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
   /* Create and display the form */
   java.awt.EventQueue.invokeLater(new Runnable() {
      public void run() {
         new JFrame().setVisible(true);
      }
   });
}

// Variables declaration - do not modify
private javax.swing.ButtonGroup buttonGroup1;
private javax.swing.JButton closeB;
private javax.swing.JPanel end;
private javax.swing.JButton homeB;
private javax.swing.JDialog jDialog1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel17;
private javax.swing.JLabel jLabel18;
private javax.swing.JLabel jLabel19;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel20;
```

```
    private javax.swing.JLabel jLabel21;
    private javax.swing.JLabel jLabel22;
    private javax.swing.JLabel jLabel23;
    private javax.swing.JLabel jLabel24;
    private javax.swing.JLabel jLabel25;
    private javax.swing.JLabel jLabel26;
    private javax.swing.JLabel jLabel27;
    private javax.swing.JLabel jLabel28;
    private javax.swing.JLabel jLabel29;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel30;
    private javax.swing.JLabel jLabel31;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JRadioButton jRadioButton3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField5;
    private javax.swing.JButton nextB;
    private javax.swing.JPanel start;
    private javax.swing.JButton startB;
    private javax.swing.JPanel step1;
    private javax.swing.JPanel step2;
    private javax.swing.JPanel step3;
    private javax.swing.JPanel task;
    private javax.swing.JButton theoryF;
    private javax.swing.JButton theoryW;
    // End of variables declaration
}
```