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

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ТРЕНАЖЕРУ З ТЕМИ «КОНТЕКСТОВІЛЬНІ ГРАМАТИКИ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ТЕОРІЯ ПРОГРАМУВАННЯ»

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Скромінський Михайло Віталійович
_____ «___»_____2020р.
(підпис)
Науковий керівник к.ф.-м.н., доц. Черненко Оксана Олексіївна
_____ «___»_____2020р.
(підпис)

ПОЛТАВА 2020р.

ЗМІСТ

ВСТУП

У сучасному і продвинутому світі Інтернет являється невід'ємною частиною нашого життя. Інтернет допомагає нам знаходити потрібну нам інформацію, проглядати різноманітні фільми, а також комп'ютерні ігри, також інтернет допомагає нам спілкуватись зі своїми друзями, рідними або з людьми, які зв'язані з роботою. Також інтернет можна використовувати для дистанційного навчання, щоб получити знання по своїй професії.

Дистанційне навчання – це доставлення певної інформації викладачами до студентів. Дистанційним навчанням може служити прямий контакт, а також і непрямий контакт. Прямий контакт може здійснюватися за допомогою таких програм як **«Skype», «Viber»** і так далі. Непрямим контактом для навчання може служити сайт, який заздалегідь підготували для навчання студентів, в якому зберігаються усі лекції, а також практичні з усіх предметів.

Метою роботи є розробка тренажера з теми «Контекстовільні граматики» для засвоєння теоретичних знань за допомогою програмних засобів Java.

Об'єктом розробки в даній роботі є процес дистанційного навчання математичним дисциплінам.

Предметом розробки є програмний продукт, що реалізує тренажер для закріплення знань із застосування контекстовільних граматик.

Головне завдання – розробити алгоритм роботи тренажеру відповідно до теми роботи та створити його програмну реалізацію.

Для реалізації тренажера з теми «Контекстовільні граматики» використано таке програмне забезпечення, як NetBeans, а також мова 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 плюси та мінуси використання дистанційного навчання, огляд тренажерів. У третьому розділі представлено основні позначення та поняття з теми, алгоритмізацію задачі за темою роботи, блок-схему, яка підлягає програмуванню. У четвертому розділі – описано обґрунтування вибору програмних засобів для реалізації завдання роботи, процес програмної реалізації, необхідну користувачу програми інструкцію.

Обсяг пояснювальної записки: 47 стор., в т.ч. основна частина - 36 стор., джерела - 10 назв.

1. ПОСТАНОВКА ЗАДАЧІ

Програма-тренажер забезпечує:

- послідовне виведення на екран завдань;
- контроль за діями користувача з розв’язання запропонованого завдання;
- миттєву реакцію на неправильні дії;
- демонстрацію правильного розв’язання завдання;
- виведення підсумкового повідомлення про результати роботи користувача.

Оцінювання навчальних досягнень є призначенням контролюючих програм. Такі програми називають також тестувальними, тому що вони здійснюють контроль на основі тестів.

Тест — це сукупність завдань спеціальної форми, призначених для перевірки засвоєння матеріалу конкретної теми або декількох тем.

Тестові завдання відрізняються від звичайних тим, що вони є короткими за змістом, вимагають чіткої відповіді та формулюються так, щоб для перевірки правильності відповіді не потрібно було аналізувати її смисл. Саме це й дозволяє використовувати комп’ютер для тестування.

Наприклад, найчастіше тестове завдання надається разом із декількома пронумерованими варіантами готових відповідей. Користувачеві потрібно вказати номер тієї відповіді, яка, на його думку, є правильною. Перевірка правильності виконання завдання зводиться до порівняння вказаного номера з відомим номером правильної відповіді.

До завдання можна надавати й декілька правильних відповідей, тоді перевіряється, чи всі вони вибрані користувачем.

Якщо завдання передбачає коротку точну відповідь, наприклад у вигляді числа або слова, то варіанти відповіді не наводяться й перевірка здійснюється як порівняння наданої й правильної відповідей — співпадають вони чи ні.

Тестування є зручним способом масової перевірки, його використовують і без комп'ютера. За єдиним для всіх тестом, за єдиними правилами, в один і той самий час визначається рівень навчальних досягнень кожного учня.

Комп'ютер дозволяє автоматизувати всі етапи тестування: програма реєструє учня, виводить на екран завдання одне за одним, приймає відповіді, перевіряє їх правильність і виводить підсумковий результат (оцінку).

Застосування комп'ютера надає тестуванню оперативності: результат тестування можна побачити на екрані одразу після виконання останнього завдання.

Комп'ютерне тестування відбувається в процесі роботи користувача з програмою, будь-яке втручання іншої людини повністю виключене. Тести, які використовуються для комп'ютерного тестування, складають досвідчені фахівці. Таким чином, комп'ютерне тестування дає нам можливість одержати об'єктивну оцінку наших досягнень.

Комп'ютер зберігає всі результати, які супроводжують перебіг тестування: список запропонованих завдань; надані відповіді; час, витрачений на кожне завдання, і т. ін. Якщо тестування проводиться систематично, то такі результати нагромаджуються й можна простежити, як прогресує учень. Отже, комп'ютерне тестування надає багато корисної інформації, воно є інформативним.

Ще одна перевага комп'ютерного тестування полягає в тому, що користувачеві, як правило, надаються всілякі зручності: для вибору відповіді достатньо вказати на неї мишею; можна змінити відповідь на іншу, відповідь приймається тільки за знаком готовності; на екрані є орієнтовна інформація — скільки часу залишилось, скільки завдань залишилось тощо.

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Плюси та мінуси використання дистанційного навчання

Зручність користування дистанційним навчанням полягає у тому, що Ви, будучи на роботі або десь на відпочинку, можете бути присутнім на лекції. Головним плюсом являється спілкування на відстані, яке дозволяє Вам у будь-якому місці, в якому є інтернет, зв’язатися з викладачем і провести лекцію або практичну.

Дистанційне навчання може дати вам більше часу тому, що Ви можете зайти на сайт і подивитися лекцію, яку випадково пропустили або не змогли на неї піти, також Ви можете виконати практичне завдання, яке також знаходиться на сайті і можете виконати його після того, як ознайомилися з лекційним матеріалом.

Дистанційне навчання також може бути зручним для тих, хто немає можливості вийти з дому таким, як діти, інваліди або тим, хто по випадковості зламав ногу, а також для тих студентів, які не мають можливості відвідувати відповідний навчальний заклад, тому що він знаходяться на великій відстані від студента і він не може його відвідувати. Також дистанційне навчання являється новинкою для сучасного навчання, воно може полегшити навчання для студентів, а також для викладачів.

Дистанційне навчання може стати постійно використовуваним для тих студентів, які не можуть відвідувати заняття по причинам, які наведені вище.

Одним із недоліків використання дистанційного навчання є те, що вчителі не можуть контролювати те, як виконується завдання студентами, тому що студенти не можуть вчасно виконати завдання.

Також постає проблема зі зв’язком через такі види зв’язку як «Skype», «Viber», тому що на великій відстані відео зв’язок може бути дуже поганим, також з’являються проблеми зі звуком.

Оскільки викладач і студенти знаходяться на великій відстані, постає проблема у тому, що викладач не має змоги перевірити практичну роботу завчасно до того, як студент її відішле. Також студенти не можуть порадитися зі викладачем щодо його предмета, з цього постає проблема у нерозумінні завдань щодо правильного виконання.

Оскільки більшість студентів не відвідують завдання, виникає проблема списування, тому що більшість завдань однакові за розв’язком, також ті студенти, які не відвідують занять не бажають виконувати завдання.

Також є легкий доступ до практичних, які деякі студенти вже зробили і які були вже оцінені викладачами і тому певні студенти можуть отримати до них доступ.

2.2. Огляд тренажерів

Самостійна робота студентів будь-яких форм навчання потребує наявності засобів, що полегшують вивчення матеріалу. Одним із інструментів при організації самостійної роботи є комп’ютерні тренажери, які все частіше застосовують в дистанційному навчанні. Тренажери призначені для вивчення і закріплення різноманітних практичних навичок.

При виконанні роботи були розглянуті деякі існуючі в ПУЕТ та інших ВНЗ тренажери з різних дисциплін.

Першим було розглянуто тренажер з теми «Верифікація програм та алгоритмічно нерозв'язні проблеми» Бернацької Вікторії дистанційного курсу «Теорія програмування» [2]. На головній сторінці надається вибір одного з трьох завдань (рис. 2.1).

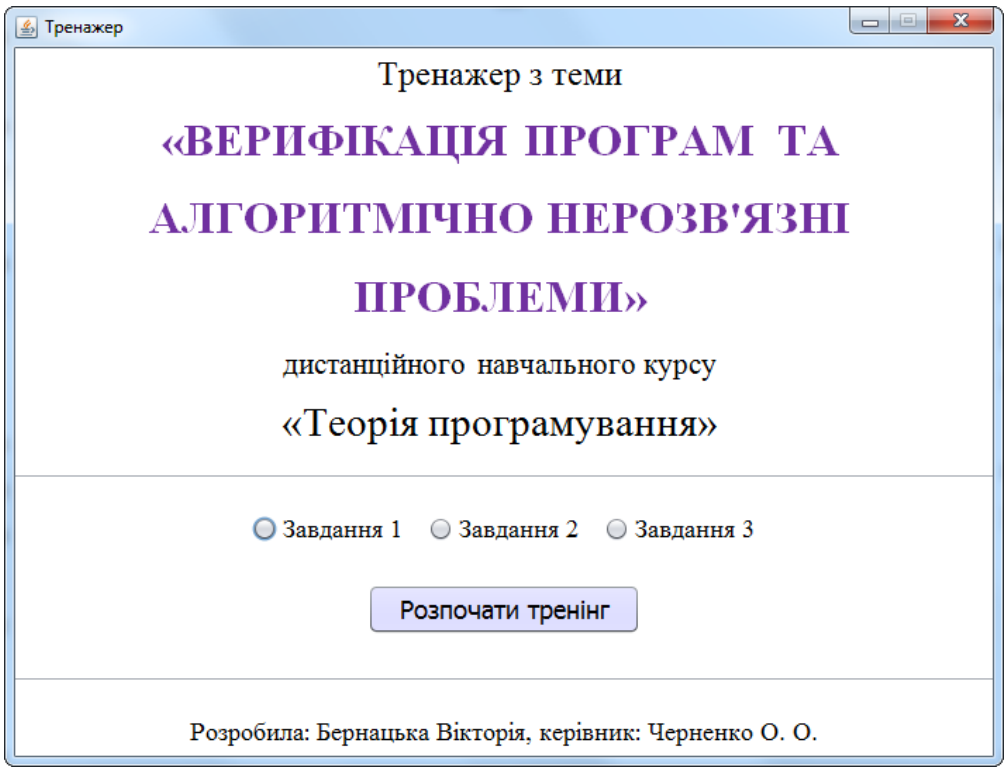

Рисунок 2.1 – Тренажер з теми «Верифікація програм та алгоритмічно нерозв'язні проблеми»

Для кожного прикладу виводиться умова і питання, потрібно вибрати відповідь (рис. 2.2).

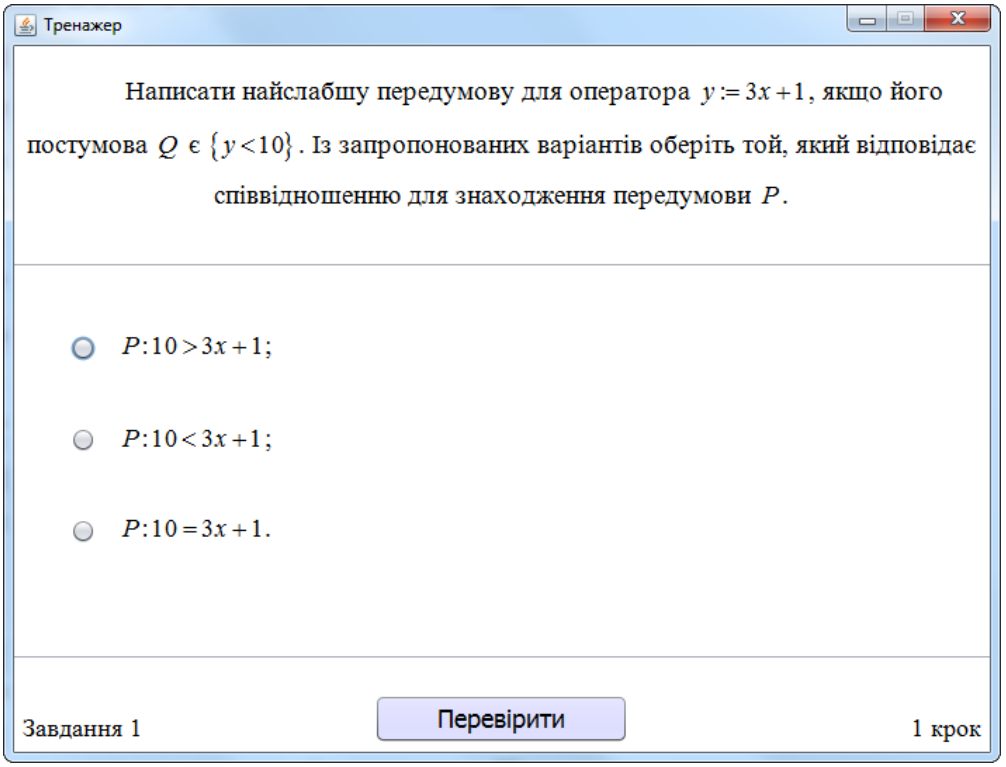

Рисунок 2.2 – Перше питання завдання 1

Якщо відповісти невірно з’явиться повідомлення з підказкою (рис. 2.3).

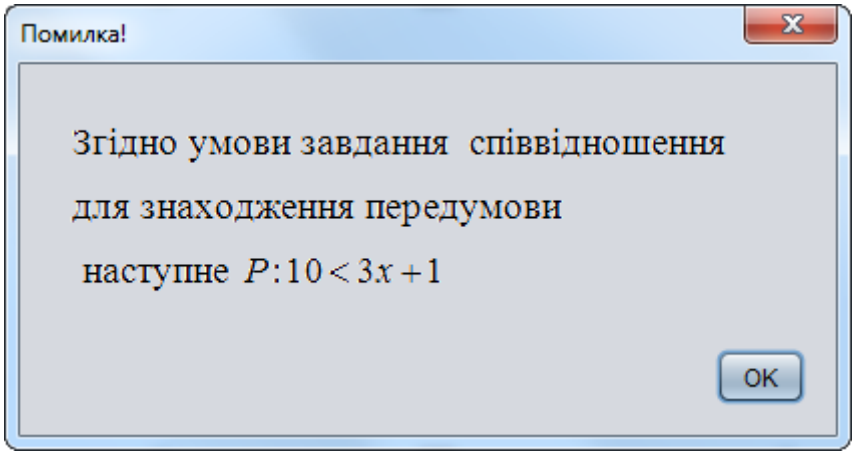

Рисунок 2.3 – Повідомлення з підказкою

Також є тести, де необхідно вказати відповідь (рис. 2.4).

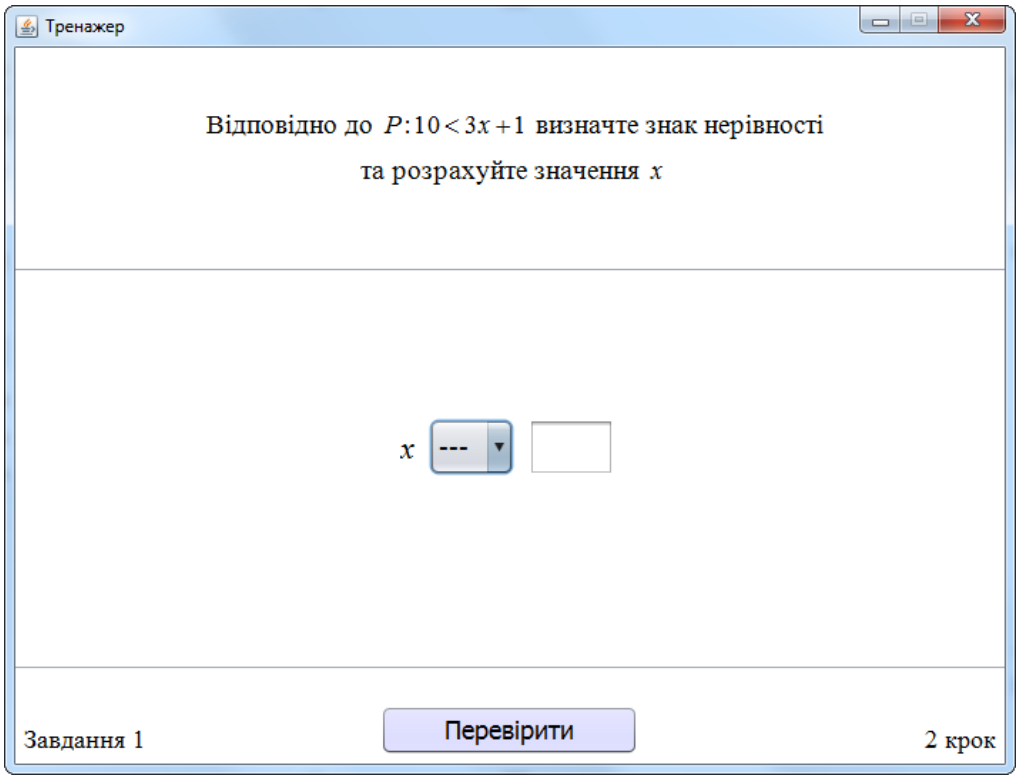

Рисунок 2.4 – Друге питання завдання 1

Інші завдання розв’язуються аналогічно розглянутому.

Наступним було розглянуто тренажер з теми «Синтаксичний аналіз» Коршун Ольги дистанційного курсу «Теорія програмування» [2]. На

головній сторінці відразу виводиться і тема, і умова прикладу, і процес розв’язання (рис. 2.5).

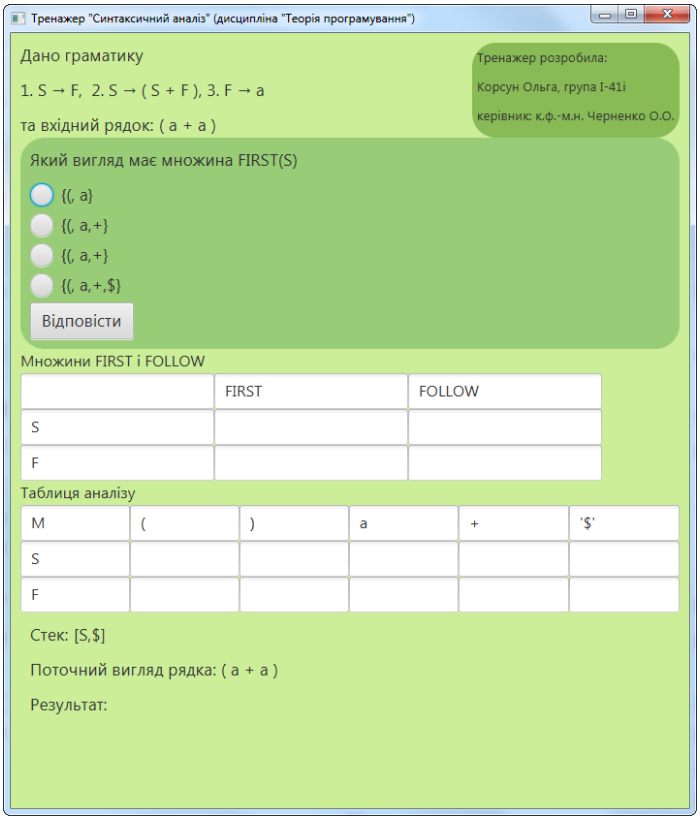

Рисунок 2.5 – Тренажер з теми «Синтаксичний аналіз»

Якщо відповідь надано невірно, то виведеться повідомлення про помилку (рис. 2.6).

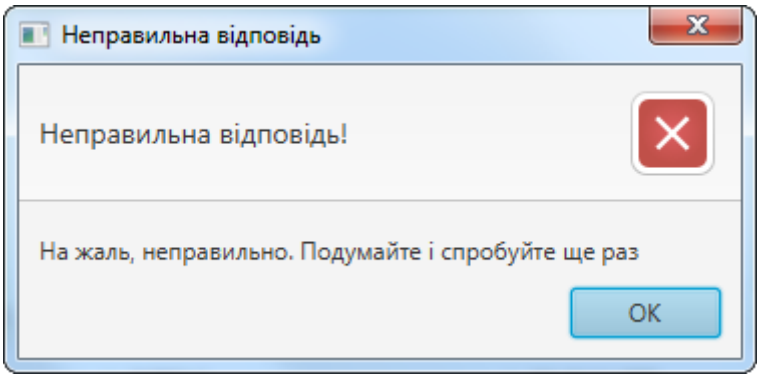

Рисунок 2.6 – Повідомлення про помилку

Під час проходження поступово заповнюються дві таблиці (рис. 2.7).

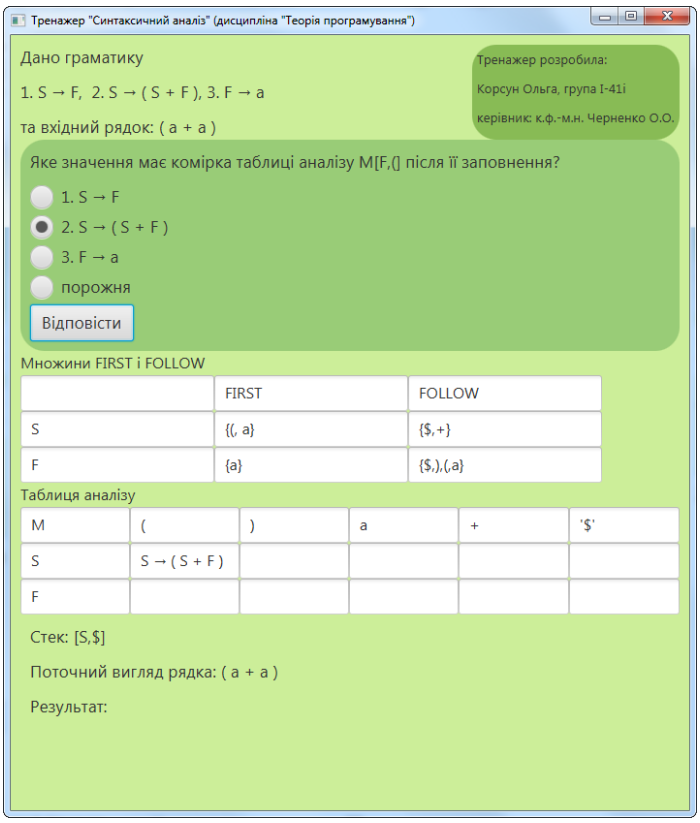

Рисунок 2.7 – Заповнення таблиць

Ще один тренажер, що розглядався – це «Машини Тюрінга» Бардаченко Світлани дистанційного курсу «Теорія алгоритмів» [3]. На головній сторінці пропонується перейти до теоретичного матеріалу або розпочати тести (рис. 2.8).

При відкритті теорії завантажується файл, в тестах є декілька варіантів відповіді:

- заповнити одну з комірок (рис. 2.9);
- вибір з варіантів відповіді (рис. 2.10);
- заповнення декількох комірок (рис. 2.11).

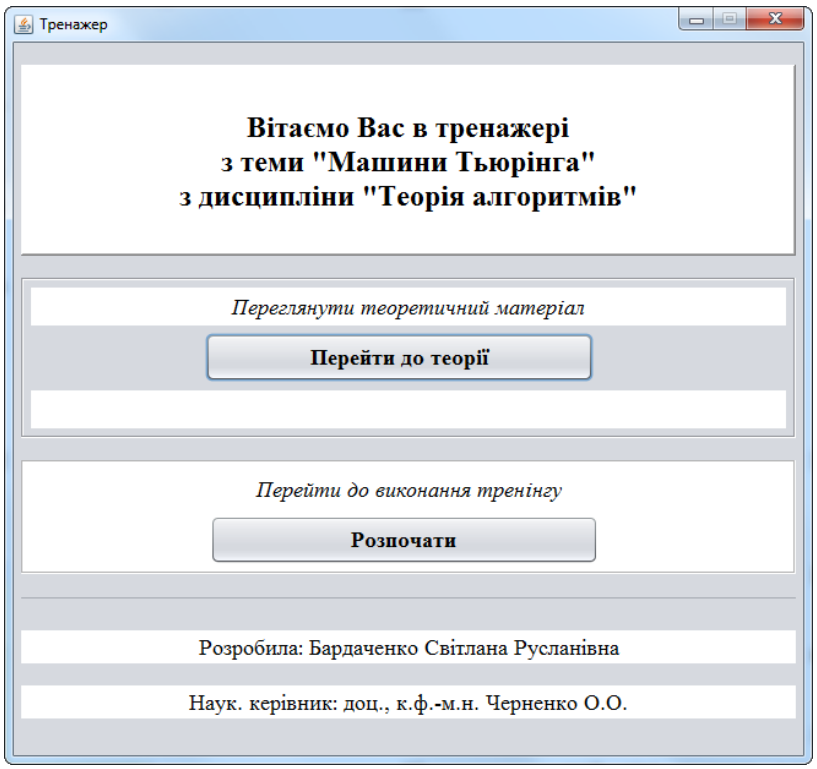

Рисунок 2.8 – Тренажер з теми «Синтаксичний аналіз»

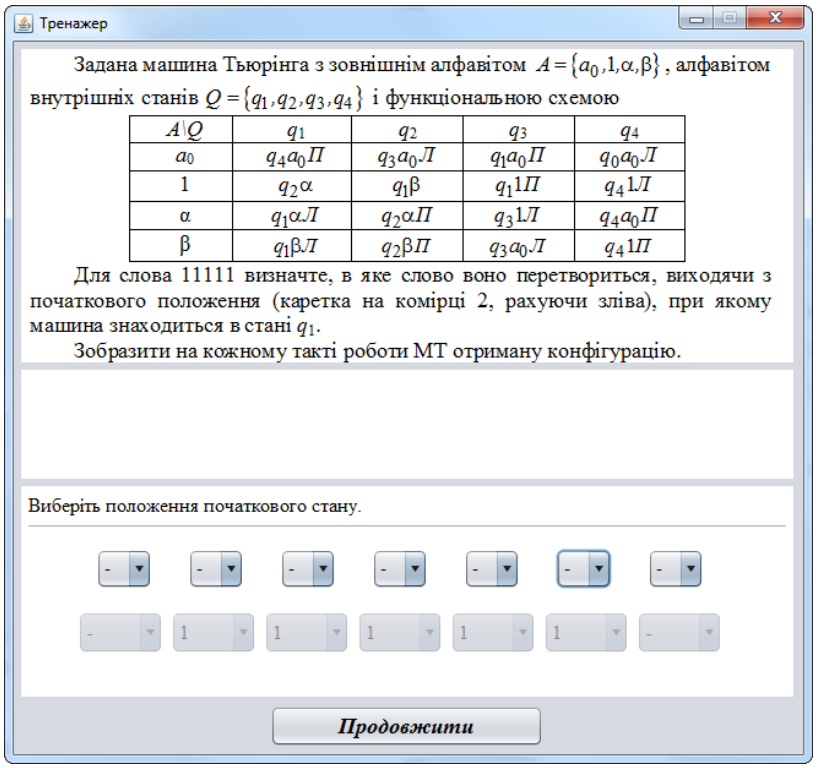

Рисунок 2.9 – Заповнення однієї з комірок

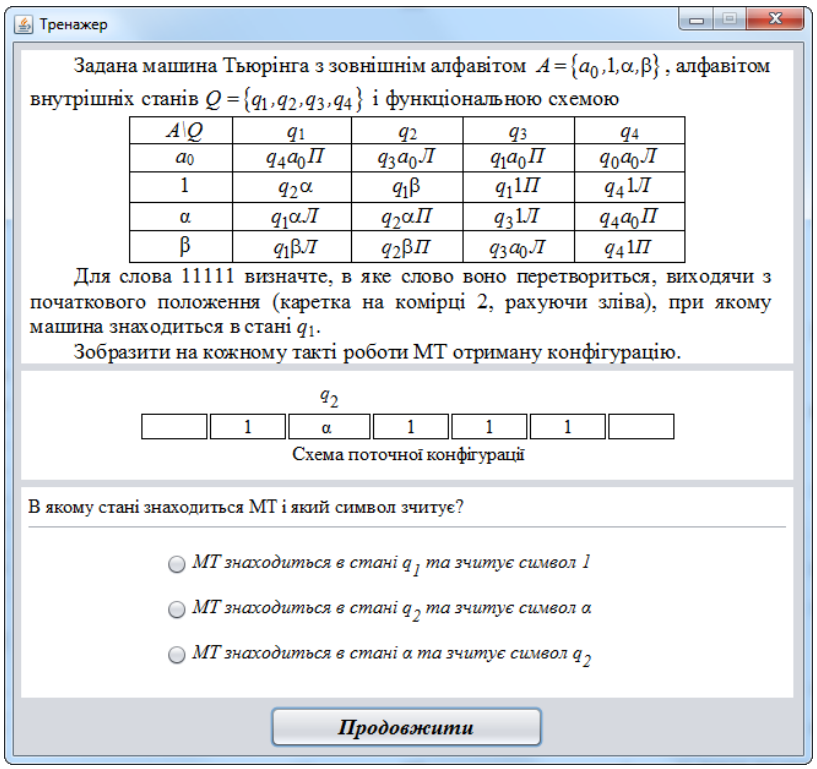

Рисунок 2.10 – Вибір з варіантів відповіді

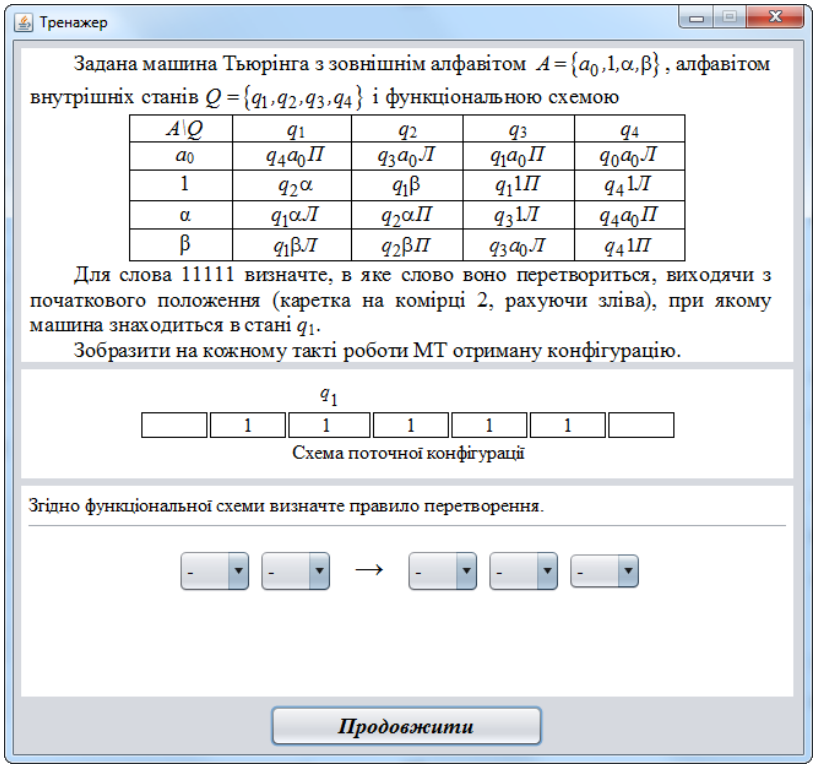

Рисунок 2.11 – Заповнення декількох комірок

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Основні позначення та поняття

Контекстно-вільною, або КВ-граматикою, називається граматика, в якій ліві частини всіх продукцій є нетерміналами. Зміст терміну "контекстно-вільна" полягає в тім, що застосування продукції A® w до ланцюжка uAv не залежить, тобто є вільним від сусідніх з A символів, які утворюють контекст uv.

Контекстовільна граматика (КВ граматика) має чотири компоненти (V_T, V_N, P, S) і задовольняє умовам:

1. у лівій частині продукції повинен знаходитися тільки один не термінал;

2. для всіх продукцій $\alpha \to \beta$ довжина рядка α, обчислена в кількості символів, буде не більше довжини рядка β.

З чотирьох типів граматик контекстно-вільні граматики (КВ) найбільш важливі з погляду додатків до мов програмування і компіляції. За допомогою КВ-граматики можна визначити велику частину структури мови програмування. При побудові граматик, що задають конструкції мов програмування, часто доводиться вдаватися до їх перетворення, щоб породжувана мова набула потрібної структури. Тому спочатку розглянемо трохи досить простих, але важливих перетворень КВ-граматик. Перший вид перетворення пов'язаний з видаленням із граматики зайвих символів. Зайві символи в граматиці можуть виявитися в таких випадках:

а) якщо символ не може бути отриманий при виведенні;

б) якщо із символу не може бути отриманий кінцевий термінальний ланцюжок (виходить нескінченний ланцюжок або немає правил, що приводять до термінального ланцюжка).

Спочатку розглянемо алгоритм виявлення символів, з яких не можна вивести кінцеві ланцюжки.

Позначене упорядковане дерево D називається деревом виводу (розбору) у КВ – граматиці (V_T, V_N, P, S), якщо виконані такі умови:

1. Кожен внутрішній вузол позначений змінною з V_T.

2. Кожна вершина (лист дерева) позначена символом з $V_T \cup V_N \cup \{\varepsilon\}$, корінь дерева позначено S.

3. Якщо дочірні вузли вузла A позначені зліва направо $X_1, X_2, ..., X_n$, то $A \to X_1, X_2, ..., X_k$ є продукцією в КВ – граматиці.

4. Якщо корінь дерева має єдиного нащадка, що позначений ε, то цей нащадок домінує над деревом, що складається з єдиної вершини ε, і $S \to \varepsilon$ - продукція з множини P.

Якщо подивитися на листи дерева розбору і виписати їх позначення зліва направо, одержимо ланцюжок, що називається кроною дерева і завжди є ланцюжком, виведенням нетермінала, що позначає корінь.

Розглянемо поняття ліве і праве породження.

Для обмеження числа виборів у процесі породження будемо заміняти крайній ліворуч (праворуч) нетермінал тілом його продукції. Таке породження будемо називати лівим (правим) і позначати через $\underset{lm}{\Rightarrow}$, від слова leftmost (зліва) ($\underset{rm}{\Rightarrow}$, від слова rightmost (справа))

Приклад контексто-вільної граматики

Візьмемо граматику для побудови виразів типової мови програмування. Обмежимося операторами “+”, “*”, а як аргументи візьмемо ідентифікатори, що починаються символами a або b, а далі йде ланцюжок з $a, b, 0, 1$.

Граматика може мати вигляд $G = (\{E, I\}, V_T, P, E)$, де

$V_T = \{+, *, (,), a, b, 0, 1\}$, а P – множина продукцій:

1) $E \to I$, 6) $I \to b$,

2) $E \to E + E$, 7) $I \to Ia$,

3) $E \to E * E$, 8) $I \to Ib$,

4) $E \to (E)$, 9) $I \to I0$,

5) $I \to a$, 10) $I \to I1$.

Приклад лівого породження:

$$E \Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow a * (E) \Rightarrow a * (E + E) \Rightarrow a * (I + E) \Rightarrow a * (a + E) \Rightarrow$$
$$\Rightarrow a * (a + I) \Rightarrow a * (a + I0) \Rightarrow a * (a + b0).$$

Праве породження:

$$E \Rightarrow E * E \Rightarrow E * (E) \Rightarrow E * (E + E) \Rightarrow E * (E + I) \Rightarrow E * (E + I0) \Rightarrow E * (E + b0) \Rightarrow$$
$$\Rightarrow E * (I + b0) \Rightarrow E * (a + b0) \Rightarrow a * (a + b0).$$

Будь-яке породження має еквівалентні ліве і праве породження.

Дерево розбору прикладу зображене на рис. 3.1.

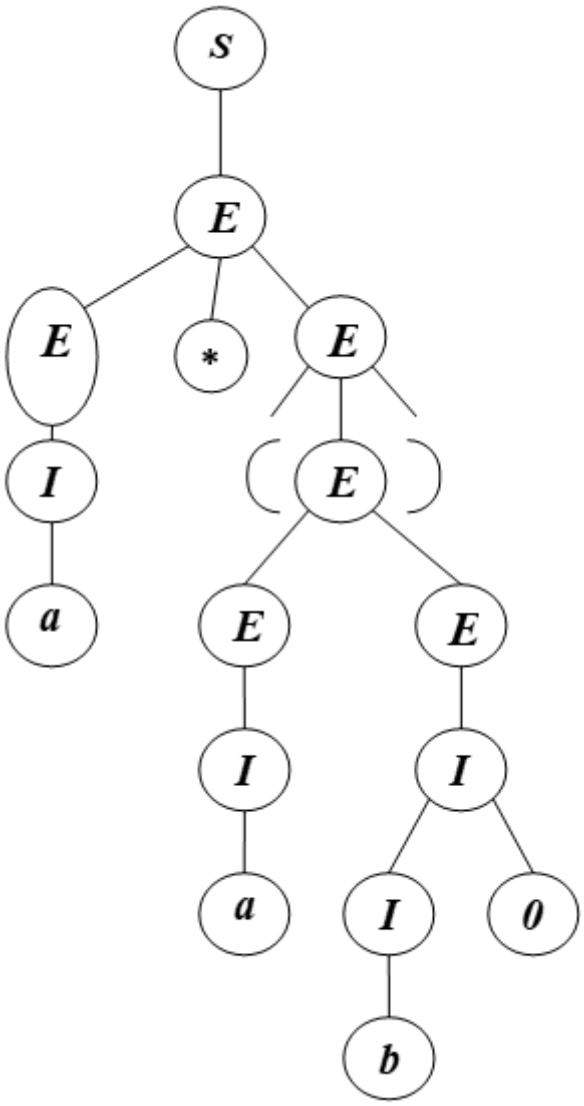

Рисунок 3.1 – Дерево розбору

Приклади

Приклад 1. Чи є задана граматика

$$G = \left(\{a,b,c\},\{S,A,B,C\},\{S \to aSBC \mid abC,\ CB \to BC, bB \to bb, bC \to bc, cC \to cc\}, S\right),$$

що задає мову $\left\{a^n b^n c^n \mid n \geq 1\right\}$ контекстно-вільною? А мова КВ?

Приклад 2. Побудувати контекстно-вільну граматику, що генерує множину ланцюжків { 0n1n | n ≥ 1}.

Приклад 3. Для граматики G_1=({+, *, ( ), a, b, 0, 1}, {E,I}, P, E), де P – множина продукцій:

1) E → I , 6) I → b,

2) E → E + E , 7) I → Ia,

3) E → E * E, 8) I → Ib,

4) E → (E), 9) I → I0,

5) I → a, 10) I → I1.

виконати наступні завдання:

1. Записати праве породження для рядка $a*(a+b0)$.

2. На прикладі генерації рядка $a+b*a$ показати, що граматика G_1 неоднозначна. Побудувати дерево розбору.

3. Граматика G_2=({+, *, ( ), a, b, 0, 1}, {E, I, T, F}, P, E), де P – множина продукцій: E → T | E + T , T → F | T * F , F → I | (E) , I → a | b | Ia | Ib | I0 | I1 породжує ту саму мову, що і G_1. На прикладі генерації рядка $a+b*a$ показати, що граматика G_2 однозначна.

Приклад 4. Розглянемо наступний фрагмент граматики для інструкцій if-then та if-then-else:

$$stmt \rightarrow if\ \ expr\ \ then\ stmt$$
$$stmt \rightarrow if\ \ expr\ \ then\ stmt\ else\ \ stmt$$
$$stmt \rightarrow other$$

На прикладі генерації рядка

if expr1 then if expr2 then stmt1 else stmt2

показати, що задана граматика неоднозначна. Побудувати однозначну граматику, в якій кожен else зв’язаний з найближчим then, з яким ще не зв’язаний вище інший else.

Заміна нетермінала з лівої частини продукції на її праву називається розгортанням нетермінала, а зворотна заміна – згортанням правої частини. Розглянемо дві стратегії аналізу, основані на згортаннях та на розгортаннях, за допомогою наступного прикладу.

Приклад 5. Нехай

G0 = ({ a, +, *, (,) }, { S, I, F },

P:

S→S-I;

$S \rightarrow I;$

$I \rightarrow I+F;$

$I \rightarrow F;$

$I \rightarrow a;$

Вони позначають вирази зі знаками операцій -, +, доданки та множники в них відповідно.

Виведення слова a-a+a в G0 з розгортанням нетерміналів, перших ліворуч у проміжних ланцюжках, має вигляд:

$S \rightarrow S\text{-}I \rightarrow I\text{-}I \rightarrow F\text{-}I \rightarrow a\text{-}I \rightarrow a\text{-}I+F \rightarrow a\text{-}F+F \rightarrow a+a+F \rightarrow$

$\rightarrow a\text{-}a+a$

Тут нетермінали, що розгортаються, підкреслені. Аналіз ланцюжка, що відтворює такі розгортання від початкового символу до термінального слова, називається низхідним, або аналізом "від верху до низу".

Тепер розглянемо виведення слова a+a*a з розгортанням нетерміналів, останніх праворуч:

$S \rightarrow S\text{-}I \rightarrow S\text{-}I+F \rightarrow S\text{-}S+a \rightarrow S\text{-}F+a \rightarrow S\text{-}a+a \rightarrow S\text{-}a+a \rightarrow S\text{-}a+a \rightarrow a\text{-}a+a$

Проміжні слова в цьому виведенні, записані у зворотному порядку, дістаються згортаннями правих частин продукцій, починаючи з термінального слова. Такі згортання від ланцюжка терміналів до початкового нетермінала граматики відтворюються в процесі висхідного аналізу, або аналізу "від низу до верху".

Головною проблемою побудови алгоритмів аналізу в обох випадках є необхідність вибору продукції, застосованої для розгортання чи згортання. Чому, наприклад, у першому виведенні на першому кроці вибирається продукція E® E+T, а не E® T, а на другому, навпаки, E® T ? Чому за оберненого виведення в слові E+T*F, в якому є дві праві частини продукцій E+T і T*F, саме ланцюжок T*F згортається в T, а не E+T в E ? Тут необхідний вибір зроблено тому, що структура термінального слова була відома заздалегідь. Але, взагалі, структура слова до початку його

аналізу невідома, і виникає необхідність перебирати продукції для застосування потрібної.

Теоретично, можна розробити алгоритм аналізу на основі перебирання продукцій, але він буде практично неприйнятним внаслідок його оцінки складності. Один із шляхів до ефективних алгоритмів аналізу полягає в обмеженні структури продукцій і позбавленні від перебирання за рахунок звуження множини КВ-граматик. Далі розглядаються саме такі обмежені граматики та побудова алгоритму аналізу для них, складність якого лінійна [3-5].

3.2. Алгоритмізація задачі за темою роботи

Користувачу відображується головна сторінка програми у якій знаходиться інформація, щодо назви тренажеру та є можливість переглянути теоретичний матеріал, а також кнопка «Start». При натиснення на неї відображається умова завдання і можливість перейти до його виконання. На кожному кроці виведено умову задачі та питання. Розглянемо алгоритм роботи тренажера:

Крок 0. Відображається умова:

Нехай G0 = ({ a, +, *, (,) }, { S, I, F }, *P:*

S→S-I;

S→I;

I →I+F;

I → F;

I →a;

Вони позначають вирази зі знаками операцій -, +, доданки та множники в них відповідно.

Кінцеве слово a-a+a. Описати розгортання нетерміналів, перших ліворуч у проміжних ланцюжках.

Умова відображатиметься на кожному кроці.

Крок 1. Відображається умова: Виходячи з вище наведеної умови, з якого нетерміналу потрібно розпочати?

1) *S;*
2) *I;*
3) *F;*

Якщо відповідь вірна, то виконується програмою перехід на наступний крок. Якщо відповідь не вірна, то виводиться повідомлення про помилку та відповідь, або підказка: «S».

Крок 2. Відображається умова: Виходячи з вище наведеної умови, який нетермінал потрібно замінити, якщо «S→»?

1) *S-I;*
2) *I+I;*
3) *F;*

Якщо відповідь вірна, то виконується програмою перехід на наступний крок. Якщо відповідь не вірна, то виводиться повідомлення про помилку та відповідь, або підказка: «*S→ S-I*».

Крок 3. Відображається умова: Виходячи з вище наведеної умови, який нетермінал потрібно замінити, якщо «*S→ S-I→*»?

1) *S → I;*
2) *I→ F;*
3) *S→ F;*

Якщо відповідь вірна, то виконується програмою перехід на наступний крок. Якщо відповідь не вірна, то виводиться повідомлення про помилку та відповідь, або підказка: «*S → I»,* що матиме вигляд «*S→ S-I→ I-I*».

Крок 4. Відображається умова: Виходячи з вище наведеної умови, який нетермінал потрібно замінити, якщо «*S→ S-I→ I –I»*?

1) *I → I;*
2) *I→ F;*
3) *I→ a;*

Якщо відповідь вірна, то виконується програмою перехід на наступний крок. Якщо відповідь не вірна, то виводиться повідомлення про помилку та відповідь, або підказка: «$I \rightarrow F$», що матиме вигляд «$S\rightarrow S\text{-}I\rightarrow I\text{-}I\rightarrow F\text{-}I$».

Крок 5. Відображається умова: Виходячи з вище наведеної умови, який нетермінал потрібно замінити, якщо «$S\rightarrow S\text{-}I\rightarrow I\text{-}I\rightarrow F\text{-}I$»?

1) $F \rightarrow I$;
2) $F\rightarrow a$;
3) $F\rightarrow I$;

Якщо відповідь вірна, то виконується програмою перехід на наступний крок. Якщо відповідь не вірна, то виводиться повідомлення про помилку та відповідь, або підказка: «$F \rightarrow a$», що матиме вигляд «$S\rightarrow S\text{-}I\rightarrow I\text{-}I\rightarrow F\text{-}I\rightarrow a\text{-}I$».

Крок 6. Відображається умова: Виходячи з вище наведеної умови, який нетермінал потрібно замінити, якщо « $S\rightarrow S\text{-}I\rightarrow I\text{-}I\rightarrow F\text{-}I\rightarrow a\text{-}I$ »?

1) $I \rightarrow I + F$;
2) $I\rightarrow I - F$;
3) $I\rightarrow a$;

Якщо відповідь вірна, то виконується програмою перехід на наступний крок. Якщо відповідь не вірна, то виводиться повідомлення про помилку та відповідь, або підказка: «$I \rightarrow I + F$», що матиме вигляд «$S\rightarrow S\text{-}I\rightarrow I\text{-}I\rightarrow F\text{-}I\rightarrow a\text{-}I\rightarrow a\text{-}I+F$».

Крок 7. Відображається умова: Виходячи з вище наведеної умови, який нетермінал потрібно замінити, якщо « $S\rightarrow S\text{-}I\rightarrow I\text{-}I\rightarrow F\text{-}I\rightarrow a\text{-}I\rightarrow a\text{-}I+F$»?

1) $I \rightarrow I + F$;
2) $I\rightarrow F$;
3) $I\rightarrow a$;

Якщо відповідь вірна, то виконується програмою перехід на наступний крок. Якщо відповідь не вірна, то виводиться повідомлення про

помилку та відповідь, або підказка: «*I → F»,* що матиме вигляд «*S→ S-I→ I-I→F-I →a-I→a-I+F→a-F+F».*

Крок 8. Відображається умова: Виходячи з вище наведеної умови, який нетермінал потрібно замінити, якщо « *S→ S-I→ I-I→F-I →a-I→ a-I+F→a-F+F »? (Літера яка підлягає під заміну підкреслена).*

1) *a* $-F+F$ на *a;*
2) *a-F+F* на *a;*
3) *a-F+F* на *a;*

Якщо відповідь вірна, то виконується програмою перехід на наступний крок. Якщо відповідь не вірна, то виводиться повідомлення про помилку та відповідь, або підказка: «*a-F+F»,* що матиме вигляд «*S→ S-I→ I-I→F-I →a-I→a-I+F→a-F+F→a-a+F».*

Крок 9. Відображається умова: Виходячи з вище наведеної умови, який не термінал потрібно замінити, якщо « *S→ S-I→ I-I→F-I →a-I→ a-I+F→a-F+F→a-a+F »? (Літера яка підлягає під заміну підкреслена).*

1) *a* $-a+F$ на *a;*
2) *a-a+F* на *a;*
3) *a-a+F* на *a;*

Якщо відповідь вірна, то виконується програмою перехід на наступний крок. Якщо відповідь не вірна, то виводиться повідомлення про помилку та відповідь, або підказка: «*a-a+F»,* що матиме вигляд «*S→ S-I→ I-I→F-I →a-I→a-I+F→a-F+F→a-a+F→a-a+a».*

Крок 10. Виводиться повідомлення про завершення проходження тренажера та кінцевий результат. Пропонується пройти тренажер знову або завершити його. Якщо вибрано повторне проходження, то відбувається перехід на крок 0.

3.3. Розробка блок-схеми, яка підлягає програмуванню

На рисунку 3.2 зображено блок-схему алгоритму роботи тренажера.

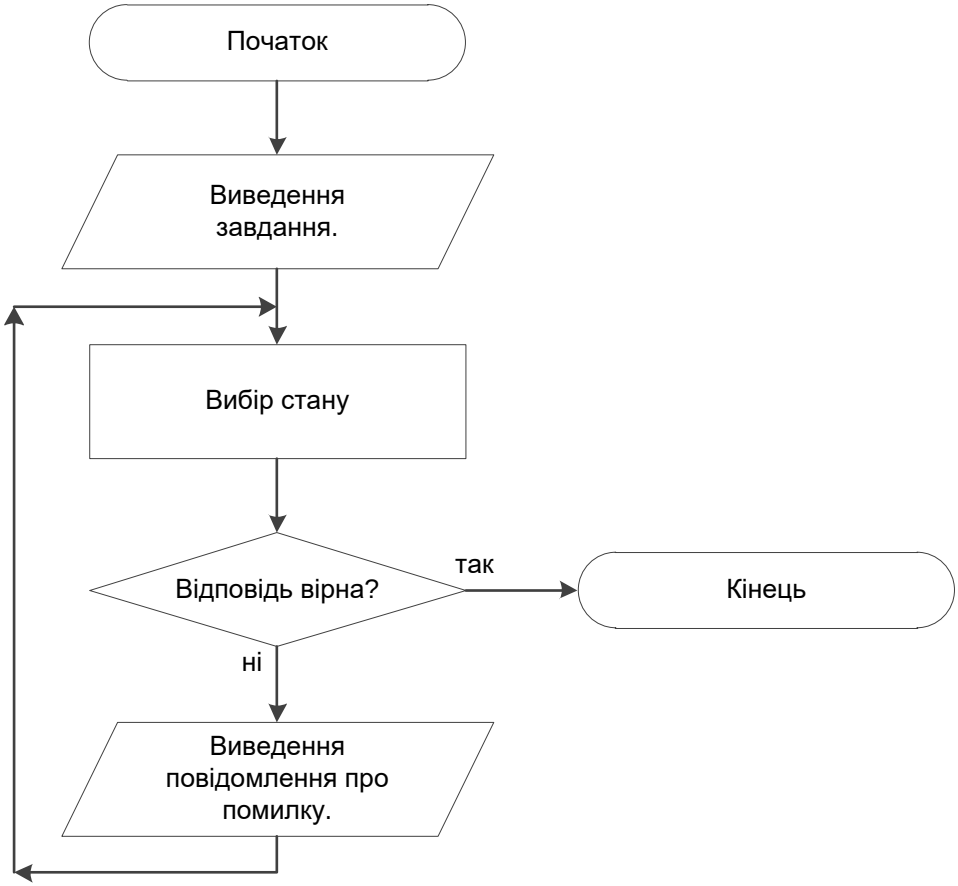

Рисунок 3.2 – Блок-схема алгоритму роботи тренажера

4. ПРАКТИЧНА ЧАСТИНА

4.1. Обґрунтування вибору програмних засобів для реалізації завдання роботи

Java є мовою програмування, за допомогою якої розробники програмного забезпечення (програмісти) створюють різни прикладні додатки для комп’ютерів, смартфонів, планшетів та інших інтелектуальних пристроїв. Особливістю програм на Java є те, що вони можуть запускатись на будь-яких комп’ютеризованих пристроях, які працюють під різними операційними системами, причому без повторної компіляції коду.

Для їх виконання необхідно лише встановити середовище для виконання – JRE (Java Runtime Environment), завантаживши його з сайту http://www.oracle.com. JRE розроблені для багатьох операційних систем – Linux(x86,x64), Mac OS X64, Solaris, Windows (x86,x64), завдяки чому код Java працює майже на всіх різновидах комп’ютерів та операційних систем.

JRE забезпечує безпечну та зручну роботу додатків на Java, тому користувачі можуть не турбуватись про несанкціоноване втручання до ресурсів свого персонального комп’ютера з боку стороннього Java коду. Необхідно лише періодично оновлювати JRE, на даний момент остання версія JRE 8 Update 171. Вбудована технологія забезпечення безпеки Java включає в себе значний набір API (Application Programming Interface) механізмів та додаткових інструментів, включаючи широковідомі та надійні алгоритми та протоколи безпеки.

Це передбачає використання криптографічних механізмів захисту інформації, інфраструктуру відкритих ключів, захищений зв'язок, автентифікацію та контроль доступу [6-7].

Крім додатків, мова Java дозволяє створювати аплети (applets). Це програми, що працюють в середовищі іншої програми - браузера. Аплету не потрібне вікно верхнього рівня - їм служить вікно браузера. Вони не

запускаються JVM — їх завантажує браузер, котрий сам запускає JVM для виконання аплету. Ці особливості відбиваються на написанні програми аплета.

З точки зору мови Java, аплет — це всяке розширення класу Applet, котрий, в свою чергу, розширяє клас Panel. Таким чином, аплет - це панель спеціального виду, контейнер для розміщення компонентів з додатковими властивостями і методами. Менеджером розміщення компонентів по замовчуванню, як і в класі Panel, служить FIowLayout. Класс Applet знаходиться в пакеті java.applet, в якому крім нього є тільки три інтерфейси, реалізовані в браузері. Треба відмітити, що не всі браузери реалізують ці інтерфейси повністю.

Оскільки JVM не запускає аплет, відпадає необхідність в методі main(), його немає в аплетах. В аплетах рідко зустрічається конструктор. Справа в тому, що при запуску першого створюється його контекст. Під час виконання конструктора контекст ще не сформований, тому не всі початкові значення вдається визначити в конструкторі. Початкові дії, зазвичай виконувані в конструкторі і методі main(), в аплеті записуються в метод init() класу Applet. Цей метод автоматично запускається виконуючою системою Java браузера зразу ж після завантаження аплета [8].

NetBeans IDE — вільне інтегроване середовище розробки (IDE) для мов програмування Java, JavaFX, C/C++, PHP, JavaScript, HTML5, Python, Groovy. Середовище може бути встановлене і для підтримки окремих мов, і у повній конфігурації. Середовище розробки NetBeans за замовчуванням підтримує розробку для платформ J2SE і J2EE.

Поширюється у сирцевих текстах під ліцензією Apache License. Проект NetBeans IDE підтримувався і спонсорувався фірмою Sun Microsystems і після придбання Sun — Oracle. У жовтні 2016 року Oracle передав NetBeans у власність Apache Software Foundation, яка займається розробкою і підтримкою проекту.

NetBeans IDE доступна для платформ Microsoft Windows, GNU/Linux, FreeBSD, і Solaris (як SPARC, так x86). Для інших платформ доступна можливість зібрати NetBeans самостійно із сирцевих текстів.

За якістю і можливостям останні версії NetBeans IDE змагається з найкращими інтегрованими середовищами розробки для мови Java, підтримуючи рефакторинг, профілювання, виділення синтаксичних конструкцій кольором, автодоповнення мовних конструкцій на льоту, шаблони коду та інше.

NetBeans IDE підтримує плагіни, дозволяючи розробникам розширювати можливості середовища [9].

4.2. Опис процесу програмної реалізації

Тренажер запускається за допомогою JApplet, вказуються його розміри.

```
public static void main(String[] args) {
    JFrame frame = new JFrame("Тренажер");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JApplet applet = new JApplet();
    frame.getContentPane().add(applet);
    applet.init();
    applet.start();
    frame.pack();
    frame.setSize(655, 470);
    frame.setResizable(false);
    frame.setLocationRelativeTo(null);
    frame.setVisible(true);
}
```

Сам аплет складається з панелей з різним розположенням. На кожній з них містяться свої елементи (рис. 4.1).

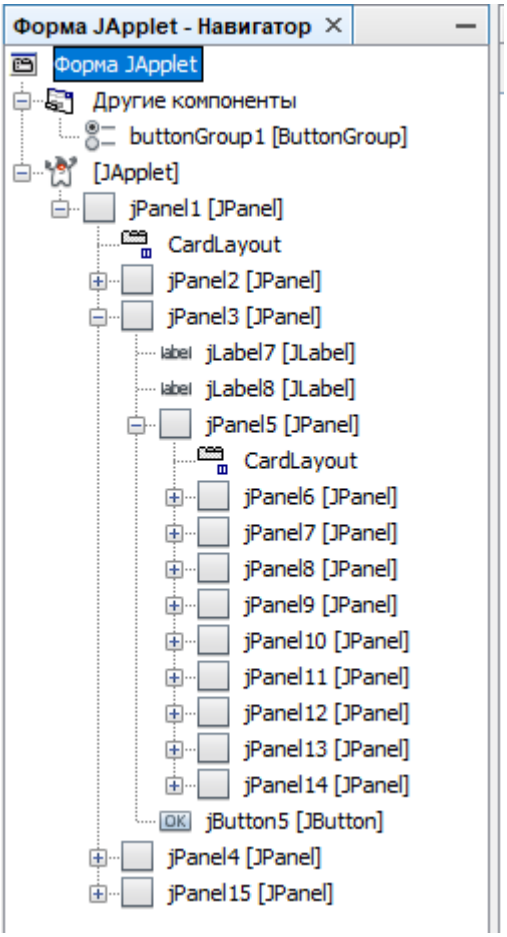

Рисунок 4.1 – Структура аплету

Щоб всі кнопки працювали, було створено події на кожну з них. Так кнопка «Теоретичний матеріал» відкриває сторінку з матеріалом.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    CardLayout cl =(CardLayout) jPanel1.getLayout();
    cl.show(jPanel1, "theory");
}
```

Для повернення від теорії до головної сторінки є «Назад».

```
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt)
{
```

```
        CardLayout cl =(CardLayout) jPanel1.getLayout();
        cl.show(jPanel1, "main");
    }
```

Кнопка «Start» відкриває тести і виводиться перше питання відповідно алгоритму.

```
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{
        CardLayout cl =(CardLayout) jPanel1.getLayout();
        cl.show(jPanel1, "example");
        cl =(CardLayout) jPanel5.getLayout();
        cl.show(jPanel5, "step1");
    }
```

Для опрацювання відповіді служить «Далі». Якщо відповідь вірна, то переключається панель на наступну, тобто виводиться інше питання. Якщо невірна, то відображається повідомлення про помилку та відповідь за допомогою JOptionPane.showMessageDialog() (див. Додаток А).

На останній сторінці є дві кнопки «Пройти знову» і «Завершити». Перша видаляє всі відповіді, що було вибрано, і встановлює перехід до умови прикладу і першого питання. Друга – закриває тренажер.

```
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{
        i = 1;
        buttonGroup1.clearSelection();
        CardLayout cl =(CardLayout) jPanel1.getLayout();
        cl.show(jPanel1, "example");
        cl =(CardLayout) jPanel5.getLayout();
        cl.show(jPanel5, "step1");
    }
```

```
        private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
{
            System.exit(0);
        }
```

4.3. Необхідна користувачу програми інструкція

На головній сторінці відображається тема тренажера, виконавець і науковий керівник (рис. 4.2).

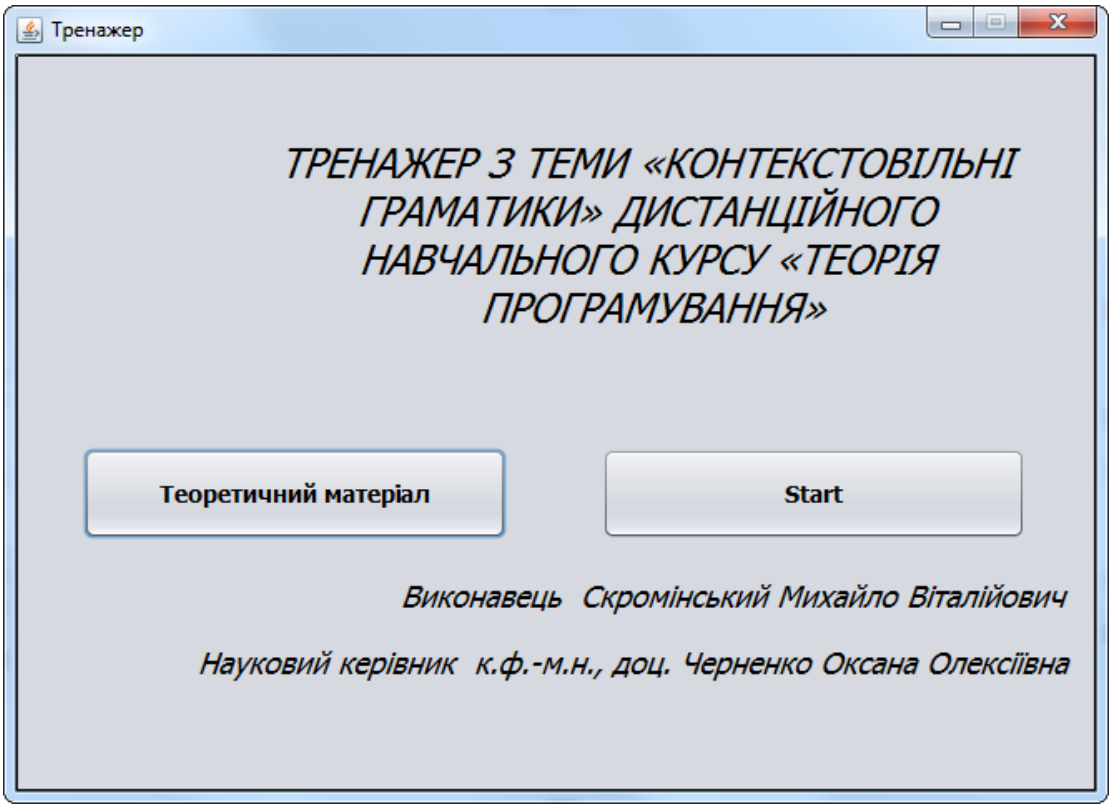

Рисунок 4.2 – Головна сторінка

Далі слід вибрати перейти до матеріалу за допомогою кнопки «Теоретичний матеріал» або розпочати тести завдяки «Start». З сторінки із матеріалом (рис. 4.3) можна повернутися до головної, натиснувши «Назад».

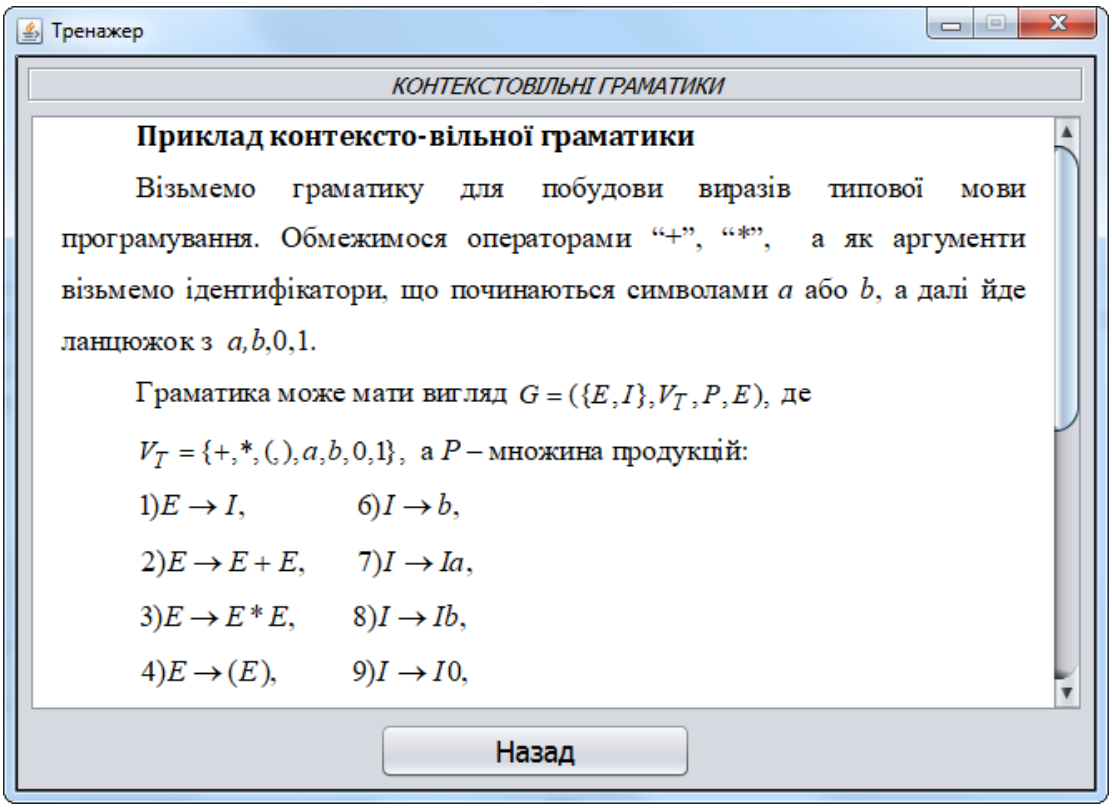

Рисунок 4.3 – Сторінка із матеріалом

В тестах на кожному кроці виводиться умова прикладу, потім задається питання і варіанти відповіді відповідно алгоритму (рис. 4.4.).

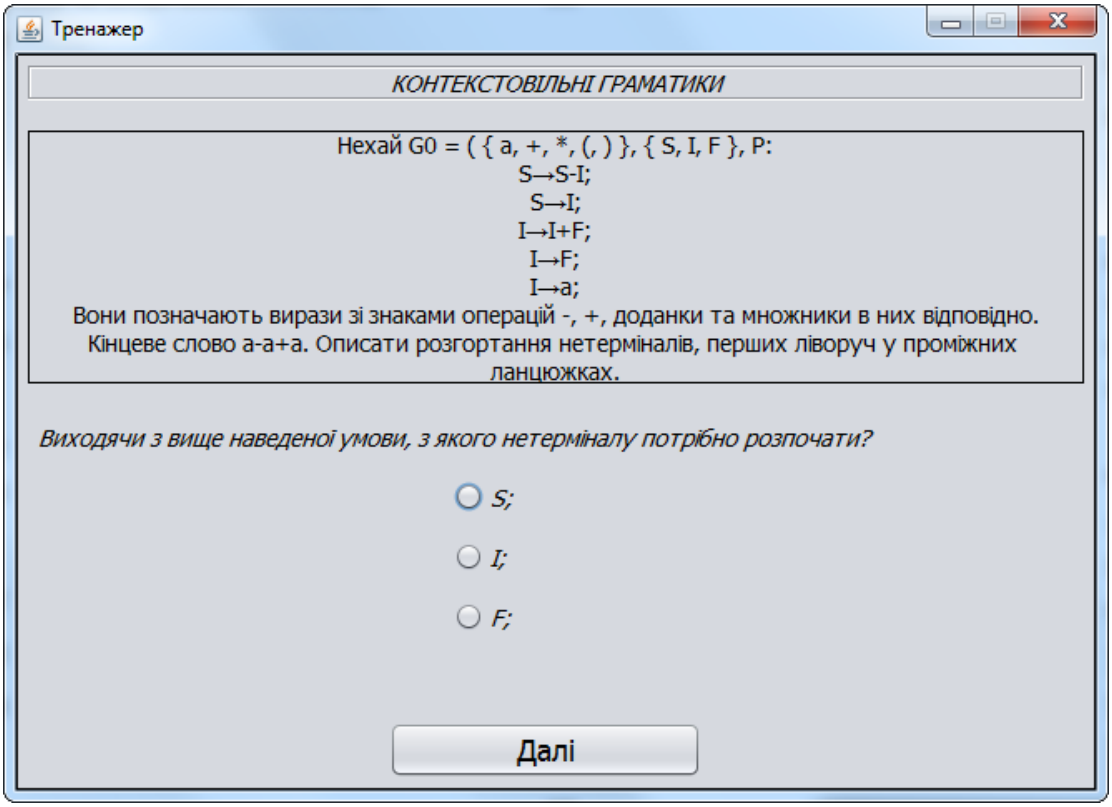

Рисунок 4.4 – Перший крок

Якщо допущено помилку, то з’явиться повідомлення про помилку та відповідь (рис. 4.5).

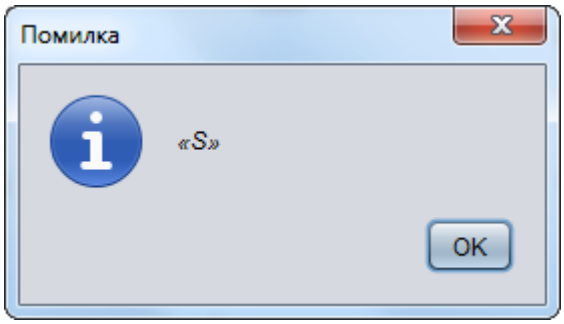

Рисунок 4.5 – Повідомлення про помилку та відповідь

На наступному кроці виведеться інше питання і варіанти відповіді (рис. 4.6).

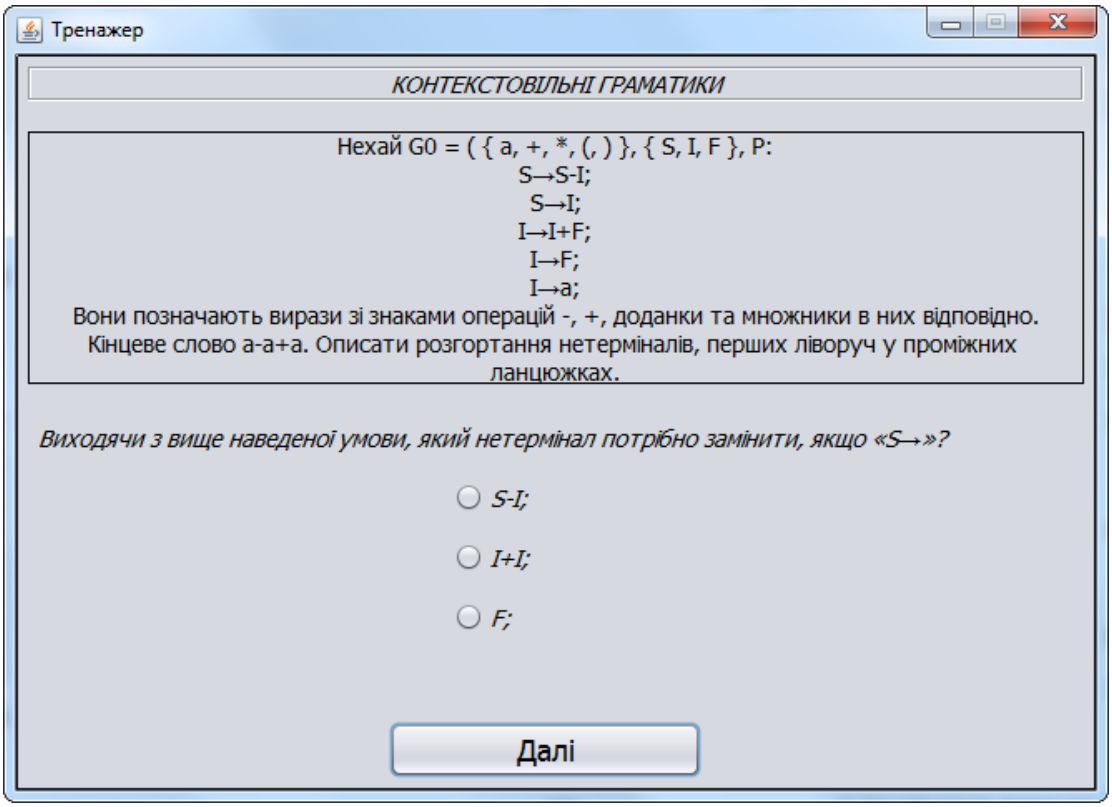

Рисунок 4.6 – Другий крок

На останньому кроці відобразиться повідомлення про завершення (рис. 4.7). Для повторного проходження слід натиснути «Пройти знову», при цьому відбувається перехід до першого кроку. Кнопка «Завершити» закриває тренажер.

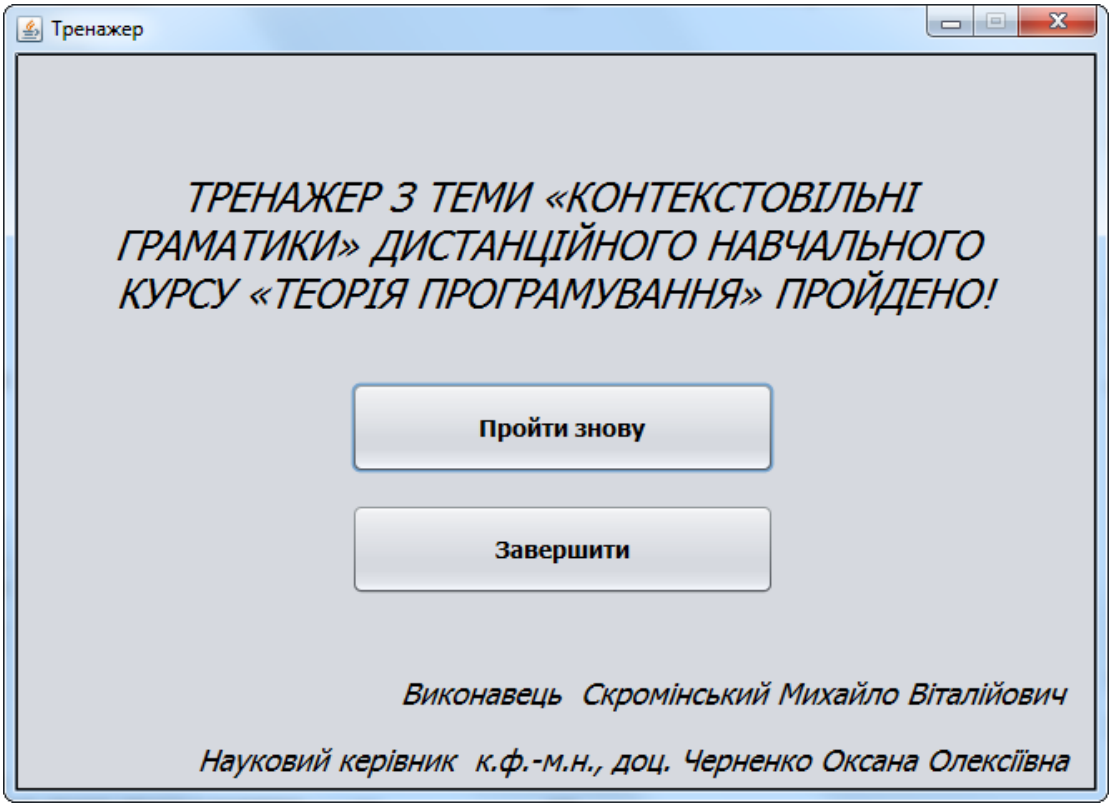

Рисунок 4.7 – Останнійкрок

ВИСНОВКИ

У даній роботі був самостійний розгляд лекційного матеріалу та засвоєння у проекті. Закріплення теоретичних знань, які були надані у лекційному матеріалі. Також був розроблений алгоритм тренажера до теми «Контекстовільні граматики» дистанційного навчального курсу «Теорія програмування».

Створення тренажерів для дистанційного навчання – це по новому відкриває навчання для студентів заочної (дистанційної) форми навчання. Перевага тренажерів в тому, що вони можуть використовуватися як для навчання студентів так і для самостійного навчання.

Основні результати роботи:

1) Розглянуто тренажери за схожою тематикою;
2) Створено теоретичний матеріал до тренажера;
3) Створено тести до тренажера;
4) Розроблено алгоритм роботи тренажера до теми «Контекстовільні граматики» дистанційного навчального курсу «Теорія програмування»;
5) Створено блок-схеми для програмування.
6) Розглянуто програмні засоби для реалізації завдання роботи;
7) Розробленно тренажер до теми «Контекстовільні граматики» дистанційного навчального курсу «Теорія програмування».

Першим було розглянуто тренажер з теми «Верифікація програм та алгоритмічно нерозв'язні проблеми» Бернацької Вікторії дистанційного курсу «Теорія програмування». Наступним було розглянуто тренажер з теми «Синтаксичний аналіз» Коршун Ольги дистанційного курсу «Теорія програмування». Ще один тренажер, що розглядався – це «Машини Тюрінга» Бардаченко Світлани дистанційного курсу «Теорія алгоритмів».

У розробленому тренажері користувачу відображується головна сторінка програми, у якій знаходиться інформація, щодо назви тренажеру

та є можливість переглянути теоретичний матеріал, а також кнопка «Start». При натиснення на неї відображається умова завдання і можливість перейти до його виконання. На кожному кроці виведено умову задачі та питання.

Програма-тренажер забезпечує:

- послідовне виведення на екран завдань;
- контроль за діями користувача з розв’язання запропонованого завдання;
- миттєву реакцію на неправильні дії;
- демонстрацію правильного розв’язання завдання;
- виведення підсумкового повідомлення про результати роботи користувача.

Результати роботи впроваджені в дистанційному навчальному 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. 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.
5. Бондаренко М.Ф. Комп'ютерна дискретна математика: Підручник / М.Ф. Бондаренко, Н.В. Білоус, А.Г. Руткас. – Харків: «Компанія СМІТ», 2004. – 408 с.
6. Мова програмування Java та платформа JavaFx: приклади застосування // Державний Університет Телекомунікацій [Електронний ресурс]. – Режим доступу: http://www.dut.edu.ua/ua/news-1-626-6031-mova-programuvannya-java-ta-platforma-javafx-prikladi-zastosuvannya
7. Шилдт Герберт. Java. Полное руководство, 8-е изд. : Пер. с англ. / Герберт Шилдт. – М. : ООО «И.Д. Вильямс», 2012. – 1104 с.
8. Java applets // Вікі ЦДПУ [Електронний ресурс]. – Режим доступу: https://wiki.cuspu.edu.ua/index.php/Java_applets
9. NetBeans // Матеріал з Вікіпедії — вільної енциклопедії [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/NetBeans

10. 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