ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ БІЗНЕСУ ТА СУЧАСНИХ ТЕХНОЛОГІЙ

ФОРМА НАВЧАННЯ ДЕННА
КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту

Завідувач кафедри _____О.О. Ємець
(підпис)

«_____»_____2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему
Алгоритмізація та програмування тренажера «Граматики. Мови, що задаються граматиками» дистанційного навчального курсу «Теорія програмування»

зі спеціальності 122 «Комп'ютерні науки»

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

ПОЛТАВА 2021 р.

ЗМІСТ

ВСТУП

Проблему впровадження інформаційних технологій в навчальний процес досліджують у великій кількості робіт. В останні роки створено непогані варіанти симуляторів широкого кола спеціальностей проте вони не узгоджені один з одним за більшістю параметрів, відрізняються операційними системами, способом подання матеріалу, їх зміст не дозволяє використовувати їх в межах єдиної освітньої програми.

Сучасні телекомунікаційні, інформаційні та комп'ютерні технології, передовсім електронні тренажери, широко використовуються для навчання. Водночас комп'ютерні технології дають змогу не тільки зменшити вартість навчання, але й покращити його якість. Адже відомо, що інформаційні технології навчання сприяють розвитку особистості студента, підготовці його до самостійної продуктивної діяльності в умовах інформаційного суспільства.

Вони передбачають (крім передачі інформації і закладених у ній знань):

- інтелектуальний розвиток (конструктивне, алгоритмічне мислення, завдяки особливостям спілкування з комп'ютером;
- креативний розвиток (творче мислення) за рахунок зменшення частки репродуктивної діяльності;
- професійний розвиток (формування уміння приймати оптимальні професійні рішення у складних ситуаціях під час комп'ютерних ділових ігор і роботи з програмами-тренажерами).

Метою роботи є алгоритмізація та програмування елементів тренажера «Граматики. Мови, що задаються граматиками» дистанційного навчального курсу «Теорія програмування».

Об'єктом розробки є процес дистанційного навчання математичним дисциплінам.

Предметом розробки є алгоритм роботи тренажеру з теми «Граматики. Мови, що задаються граматиками».

Перелік використаних методів полягає в застосуванні граматик, мов, що задаються граматиками, мова програмування Java.

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

Робота складається з чотирьох розділів. В першому розділі представлено постановку задачі. В другому розділі описано огляд комп'ютерних тренажерів, систем, середовищ програмування, середовищ для розробки програмного забезпечення, необхідність та актуальність теми роботи. В третьому розділі описано огляд матеріалу за темою роботи, алгоритмізацію задачі за темою роботи, розробку блок-схеми, яка підлягає програмуванню. В четвертому – представлено обґрунтування вибору програмних засобів для реалізації завдання роботи, опис процесу програмної реалізації, опис роботи програми.

Обсяг пояснювальної записки: 60 стор., в т.ч. основна частина - 42 стор., джерела - 14 назв.

1. ПОСТАНОВКА ЗАДАЧІ

Рішення будь-якої прикладної задачі з використанням електронних обчислювальних машин (ЕОМ) складається з наступних етапів:

- аналіз вимог і формальна постановка задачі;
- вибір або розробка математичної моделі;
- аналіз способів вирішення;
- логічне проектування і розробка алгоритму;
- кодування (написання програми);
- тестування і налагодження програмного забезпечення;
- впровадження, використання і супровід програмного забезпечення.

Постановка завдання розробки програмного забезпечення є найпершим і найбільш важливим етапом при проектуванні програмного забезпечення. Саме тут закладається фундамент майбутньої програми. Якщо на цьому етапі допущені помилки або не передбачені якісь важливі деталі, то це відіб'ється на кінцевому результаті, а вносити зміни і виправлення в такий проект буде вкрай проблематично. Більш того, якщо завдання поставлене невірно, то її подальше рішення не має сенсу. На етапі постановки завдання необхідно відповісти на наступні питання:

- чи існують методи або способи вирішення завдання без використання ЕОМ, чи потрібно вирішувати це завдання на ЕОМ;
- які «дивіденди» принесе використання ЕОМ для вирішення завдання, що буде покращено, прискорене, оптимізовано, зекономлено при використанні ЕОМ, яка основна мета застосування ЕОМ;
- що необхідно з обладнання, крім універсальної ЕОМ, який тип ЕОМ необхідний для вирішення завдання, яка платформа (и) ЕОМ може підійти, яка операційна система буде керувати роботою ЕОМ, яке додаткове програмне забезпечення потрібно;

- які бізнес-процеси і документообіг прикладної предметної області, де буде застосовуватися розробляється програмне забезпечення;
- який формат вихідних (вхідних) даних, які дані і в якій формі необхідні для вирішення завдання;
- який формат проміжних і вихідних даних, які дані і в якій формі необхідно отримати;
- який інтерфейс користувача програмного забезпечення потрібно забезпечити, який інтерфейс з додатковим обладнанням необхідний;
- яка «глибина» опрацювання користувальницької, інженерної, програмної і конструкторської документації, експлуатаційних документів, методичних посібників і керівництв по використанню програми, хто і як буде застосовувати результати рішення.

Перший крок у проектуванні програмного забезпечення полягає в точному формулюванні цілей впровадження програми. Необхідно, щоб цей процес був гнучко організований і тривав протягом тривалого часу, оскільки на будь-якому етапі розробки або впровадження можуть розкриватися раніше непередбачені проблеми, які потребують внесення в проект певних змін. У той же час для спрощення розробки слід заборонити радикальний перегляд вимог на стадії кодування (реалізації) програми. Необхідно також заздалегідь визначити умови внесення несуттєвих змін в проект. Всі домовленості такого плану повинні оформлятися розробником і замовником програми в офіційному порядку у вигляді окремих пунктів у договорі на розробку програмного забезпечення, додаткових протоколів чи актів.

З офіційним висновком договору зазвичай передують:

- з'ясування реальної необхідності такої системи;
- оцінка можливості її розробки і зразкового обсягу витрат;
- визначення очікуваного ефекту від впровадження.

Після завершення етапу попередніх досліджень складають список вимог, що пред'являються до програмного забезпечення. Сюди входять:

- аналіз обстановки (сукупність умов, в яких передбачається експлуатувати програмну систему);
- опис виконуваних програмою системою функцій (чіткий опис того, що повинна робити система, на підставі яких вхідних даних, які дані є вихідними);
- обмеження, які повинні враховуватися в процесі проектування (терміни завершення; ресурси, наявні в наявності).

Головна мета на етапі аналізу формальної постановки задачі і складання вимог до програми полягає в пошуку і поданні таких ситуацій, які можуть привести до збою в роботі програми і у визначенні причин і способів подолання таких ситуацій.

Слід розглянути приклади і використати при реалізації роботи.

Приклад 1. Мова $\{x^n y^n \mid n > 0\}$ описується граматикою

$G_1 = (\{x, y\}, \{S\}, P, S)$.

Вказати P.

Приклад 2. Граматикою для мови $\{x^m y^n \mid m, n \geq 0\}$ є $G_2 = (\{x, y\}, \{S, B\}, P, S)$.

Тут набір продукцій P має вигляд

$S \to xS$, $S \to y$,

$S \to yB$, $B \to yB$,

$S \to x$, $B \to y$.

Оскільки порожній рядок також належить мові, у набір P також входить продукція $S \to \varepsilon$.

Як генерується рядок $xxyyy$?

Приклад 3. Як можна переписати множину продукцій граматики $G_1 = (\{a,1,2\},\{A,B,C,D\},\ \{A \to BC, A \to BD, A \to B, B \to a, C \to 1, D \to 2\}, A)$?

Приклад 4. Яку мову задає граматика G_1 із прикладу 3 ?

Приклад 5. Яка граматика породжує множину ідентифікаторів?

Приклад 6. Яка граматика задає множину "вкладених дужок" $\{(^n)^n \mid n \geq 0\}$?

Приклад 7. Яка граматика визначає мову $\{a^n b^n c^n \mid n \geq 1\}$?

2. ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд комп'ютерних тренажерів

Програмними засобами навчального призначення називають цілий ряд програм, які безпосередньо призначені для забезпечення навчального процесу. Мультимедійні технології - це система комплексної взаємодії візуальних і аудіо ефектів під управлінням інтерактивного програмного забезпечення з використанням сучасних технічних і програмних засобів, які об’єднують текст, звук, графіку, фото, відео тощо в одному цифровому відтворенні

До групи засобів з елементами штучного інтелекту відповідно відносять:

- системи комп’ютерного тестування;
- комп’ютерні тренажери;
- системи навчального діалогу, та ін.

Група інших засобів складається з наступних видів:

- навчальні бази даних;
- мультимедійні довідники та енциклопедії;
- електронні підручники;
- віртуальні лабораторії, та ін.

Програмні засоби навчання відповідають вимогам сучасного часу, надаючи змогу комплексного підходу до покращення освітнього процесу практично за будь-яким напрямком, заощаджуючи час та забезпечуючи високий рівень засвоєння матеріалу. Розглянемо можливості окремих видів програмних засобів [2].

Системи комп'ютерного тестування Це програмні системи, що дозволяють проводити аналіз знань студентів за допомогою сучасних інформаційних технологій. Можна виділити наступні типи систем

комп’ютерного тестування:- за можливістю поповнення бази запитань та внесення окремих змін до програмної оболонки (відкриті, закриті);

- за інтерфейсом взаємодії з користувачем (гнучкі, формалізовані);

- за середовищем розташування (локальні, глобальні);

- за предметно-галузевою спрямованістю(профільні, загальні).

Система комп’ютерного тестування складається з наступних частин:

- оболонка тестування;

- статистичних база даних;

- система створення та підготовки тестів;

- база даних запитань з доступом до статистичної бази даних;

- база користувачів;

- система керування користувачами.

Ефективність системи комп’ютерного тестування можна підвищити за допомогою автоматизованої системи обробки та оцінювання результатів тестування [3].

Електронні бібліотеки, а також електронні бази навчально-методичної літератури, що створені у окремих структурних підрозділах освітніх установ, дають змогу працювати з ними будь-де, за наявності Інтернет з’єднання. Або ж є можливість закачати електронні підручники на будь який носій (флеш-диск, телефон, планшет, ноутбук та ін.) у разі відсутності Інтернету. До переваг електронних підручників також відносяться:

- доступність великої кількості різноманітних підручників, навчально

-методичної літератури за обраною предметно–галузевою спрямованістю;

- безкоштовність, за винятком невеликої кількості специфічних видань;

- особисті налаштування при навчанні по електронному підручнику, наприклад, розмір та тип шрифту;

- електронний підручник, на відміну від паперового, не зношується;

- якщо розглядати мультимедійний електронний підручник, то можна казати ще й за підвищення якості навчання у зрівнянні з паперовим [2].

Комп'ютерні тренажери (вільного некомерційного використання) формують практичні уміння і навички застосування набутих знань. Наприклад, сьогодні ефективними є такі математичні тренажери: Динамічна геометрія, Functor, Graphics, GrapWin, Poly (геом. просторові фігури, розгортки, правильні многогранники), Чарт, Flat Graph, Algebry, Discriminant, Gauss, GaussWin, Gorner, InFunction, Krug, KvadYr, Primer6 (дії з дес. дробами), Sistema koordinat, Математика – тренажер арифметичних дій, Математика – тренажер добування квадратного кореня тощо. Програми-тренажери забезпечують: послідовне виведення на екран завдань заданої складності з вибраної теми; контроль за діями користувача з розв'язання запропонованого завдання; миттєву реакцію на неправильні дії; виправлення помилок користувача; демонстрацію правильного розв'язання завдання; виведення підсумкового повідомлення про результати роботи користувача (можливо, з рекомендаціями чи порадами). Правильно підібрані та використані комп'ютерні тренажери не тільки підвищать рівень знань, але і допоможуть зацікавити студента відповідними дисциплінами. При роботі з програмою–тренажером кожний студент підпадає під пильне «око» комп'ютера, який виправляє його помилки і не виводить оцінку в журнал, а надає можливість удосконалювати навички до бажаного рівня.

До складу мультимедійного інтерактивного комплексу, як правило, входять:

- інтерактивна дошка з електронними олівцями;

- мультимедійний проектор;- комп'ютер викладача;

- пристрої зв'язку (веб-камера, система передачі даних, адаптер, тощо);

- спеціалізоване навчальне програмне забезпечення, методичні матеріали;

- лабораторне та демонстраційне обладнання.

До складу комплексу може також входити пристрій тактильного введення даних (інтерактивний безпровідний планшет; інтерактивний рідинокристалічний дисплей (інтерактивна графічна панель), об’єднуючий в собі функції монітора і цифрового планшета; система інтерактивного опитування – пульти, безпровідні мікрофонні системи) і система звукового супроводу [3].

Зупинимося докладніше на динамічних тренажерах, як найцікавіших і складніших програмних продуктах. Є два різні способи виготовлення таких тренажерів. Перший полягає в написанні окремої програми для кожного окремого тренажера, другої, - у використанні спеціального інструменту розробника , який дозволяє в багато разів прискорити розробку. У першому випадку можливе досягнення красивих спеціальних ефектів, але дуже утруднена модифікація тренажера. Як правило, програми цього класу (і математичні моделі в них) дуже прості, тому такі локальні тренажери поширеніші (що є показником не якості, а рентабельності їх виробництва). При використанні конструктора розробником тренажера повинні бути не програміст, а технолог, що володіє апаратом прикладної математики. Загальна властивість більшості конструкторів - складання динамічної моделі з "кубиків" - стандартних елементів, що описують певні об'єкти управління (прилади) і стандартні математичні операції, передавальні функції, логіку. Конструктори рідко і не дуже охоче пропонуються на продаж по причинах дуже високої вартості розробки, малого тиражу і, отже, високих цін. З іншого боку, при перспективному плануванні комп'ютеризації підготовки персоналу має сенс піти на підвищені витрати, щоб має можливість редагувати наявні моделі і створювати нові за власним розсудом [4].

При виборі конструктора слід враховувати:

- набір стандартних елементів і можливість його розширення;
- можливі режими роботи і розв'язування завдання (показ правильних дій, сценарії аварійних ситуацій, оцінка навчаного);
- якість і ергономічні характеристики інтерфейсу (зручність роботи) розробника і навчаного.

Ряд колективів використовують (і пропонують на продаж) конструктори динамічних тренажерів більш менш прийнятної якості. Наш конструктор виконаний за об'єктно-орієнтованою технологією і забезпечує зручний графічний інтерфейс як для навчання, так і для розроблення. Моделі складаються із стандартних об'єктів, кожен з яких має своє зображення, механізм управління і спосіб моделювання. Набір елементів за бажанням замовника легко може бути розширений. Закінчена модель є набором з декількох вікон, об'єднаних один з одним за ієрархічним принципом, або за принципом циклічного списку. У кожен окремий момент на екрані може бути видно одне або декілька вікон. Частину вікон з керованими об'єктами складають інтерфейс навчання. Управління моделлю так само легко здійснюється як за допомогою миші, так і з клавіатури. У недоступних для навчання вікнах розробник збирає динамічну модель, маючи для цього багатий набір стандартних елементів [5].

2.2. Системи, середовища програмування, середовища для розробки програмного забезпечення

Застосування немашинних мов – асемблерів, мов високого та дуже високого рівня – дало поштовх до розробки інструментів, що забезпечують автоматичне складання машинних програм для комп’ютерів на основі інформації, поданої у програмах, написаних немашинною мовою. Ці

інструменти називаються засобами (системами) автоматизації програмування.

Системи автоматизації програмування (системи програмування) спочатку було зорієнтовано на мови FORTRAN, ALGOL, COBOL і являли собою програми перекладу немашинних (вхідних) програм у машинні (вихідні).

Системи автоматизації програмування можуть бути залежними від однієї з вихідних мов або зорієнтованими на клас мов. В останньому разі вони називаються параметричними.

Розрізняють два типи систем автоматизації програмування: найпростіші, що працюють за принципом „перетворити-завантажити-виконати" і асемблерні, які працюють за принципом „перетворити-зібрати-завантажити-виконати".

Результат роботи системи програмування першого типу – програма машинною мовою в абсолютних адресах, готова до виконання.

Результат роботи системи другого типу – „напівскомпільована" програма машинною мовою, побудована з використанням відносних адрес. Ця програма не готова до виконання. Щоб виконати таку програму необхідно використати завантажувач, який перетворює в програмі відносні адреси в абсолютні.

Завантажувачі – це програми, які виконують дві такі функції:

- зв'язують (збирають) основну програму з інших програм, використаними в основній;
- за допомогою абсолютних адрес налаштовують (завантажують) складну програму на конкретне місце в пам'яті.

Іноді ці функції виконуються двома різними засобами – редактором зв'язків і завантажувачем.

Асемблерні системи програмування набули подальшого розвитку в середовищах програмування завдяки тому, що дають змогу:

- об’єднати напівкомпільовані програми;
- об’єднувати програми написані різними мовами;
- створювати різні конфігурації машинної програми з одних і тих самих напівкомпільованих частин.

Нині системи автоматизації програмування в описуваному вигляді не застосовуються. Проте вони становлять основу засобів програмування, що поділяються на такі типи: середовища програмування і середовища для розробки програмного забезпечення [6].

Середовища програмування (programming environment) – це системи автоматизації програмування, які забезпечують створення, програмування, кодування та налагодження програм. Крім засобів трансляції програм вони містять засоби роботи з файловою системою операційних систем, редактор для створення програм на мові програмування, налагоджувач, який забезпечує різні режими виконання програми, а також розвинені бібліотеки підпрограм, макросів, модулів, класів, мегамодулів. Іноді їх називають інтегрованими середовищами розробки (програмування) (Integrated Development Environment - IDE).

Середовища розробки програмного забезпечення (Computer Aided Software Environment CASE) – це купа засобів програмування, зорієнтованих на виконання не лише вертикальних програмних процесів, а й горизонтальних. Вони забезпечують наступне:

- повну підтримку життєвого циклу;
- повну інтеграцію засобів в аспекті колективної роботи;
- загальні репозитарії (данні, документація, продукти фаз життєвого циклу);
- стандартний інтерфейс, що не залежить від фази життєвого циклу;
- апаратну та програмну переносність (мультиплатформеність);
- підтримку мережної колективної роботи [7].

2.3. Необхідність та актуальність теми роботи

Дистанційна освіта і до карантину з самоізоляцією була дуже популярною з багатьох причин. На сьогоднішній день невідомо, скільки ще часу триватиме пандемія, а вчитися треба завжди. Тому дистанційній формат буде актуальний ще довгий час. На дистанційній формі навчання можна здобути вищу освіту, середньо-спеціальну, а також пройти курси перепідготовки. Розглянемо переваги такого формату.

У кожному місті чи населеному пункті є свої навчальні заклади, і в них зустрічається якісна освіта. Але багато розташовані далеко або в інших містах. Можливо, людина хоче отримати професію, якої не має в його місті чи державі. Тому дистанційна форма - це прекрасний варіант вчитися навіть за кордоном. Людина залишається вдома на своєму місці, тому немає паперової тяганини і труднощів, пов'язаних з переїздом. Дистанційна форма відрізняється від заочної тим, що заняття відбуваються постійно, є взаємодія учня і викладача, інтенсивний контроль знань.

Дистанційне навчання особливо підходить тим, хто вже працює і хоче отримати чергову освіту і підвищити рівень професіоналізму. Тобто підходить тим, хто не може очно перебувати в закладі і хоче отримати знання, а не тільки диплом або сертифікат. Найчастіше дистанційний формат передбачає гнучкий графік, коли людина вивчає матеріал тоді, коли йому зручно. Можна виділити час після роботи, у вихідні дні і навіть на перервах, при поїздках в транспорті.

За ціною дистанційна освіта часто дешевше, ніж звичайний формат. Очне навчання з відвідуванням навчального закладу часто багатьом просто недоступно з матеріальних причин.

У реальному житті існують навчальні заклади, які спеціалізуються на певній вузькій галузі. При такій спеціалізації викладачі дуже добре розбираються в матеріалі саме в своїй сфері. Тому педагогів привертають

для проведення дистанційних занять. Це дозволяє людині вчитися у самих грамотних педагогів з усього світу.

«Дуже серйозною проблемою дистанційного навчання є переосмислення використання багатьох перевірених педагогічних прийомів для кращого запам'ятовування і засвоєння матеріалу, таких, як: метод опорних точок, метод свідомих помилок, метод вибору кращого рішення і т. Д. Застосування самих різних педагогічних методів стає більшою мірою залежним від технічних засобів і способів організації контакту з учнями. Однак необхідно відзначити при будь-якій технології взаємодії викладачеві доводиться вчитися більш стисло і чітко викладати матеріал або відповідати на питання. І в даній ситуації стає концептуальним постійне і безперервне самовдосконалення як викладача, так і того, хто навчається» [8].

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Огляд матеріалу за темою роботи

Формальна граматика або просто граматика в теорії формальних мов — спосіб опису формальної мови, тобто виділення деякої підмножини з множини всіх слів деякого скінченного алфавіту. Розрізняють породжувальні і аналітичні граматики — перші ставлять правила, за допомогою яких можна побудувати будь-яке слово мови, а другі дозволяють по даному слову визначити, входить воно в мову чи ні. Формальні граматики були введені американським вченим, математиком та філософом, Н. Хомським у 50-тих роках XX сторіччя.

Розглянемо поняття граматика. **Граматика** має чотири компоненти:

1. Безліч термінальних символів, іноді іменованих токенів. Термінали представляють собою елементарні символи мови, з яких формуються рядки.

2. Безліч нетерміналів, які іноді називають синтаксичними змінними. Кожен нетермінал – це безліч рядків терміналів .

3. Безліч продукцій, кожна з яких складається з нетермінала, який називають заголовком або лівою частиною продукції, стрілки і послідовності терміналів і / або нетерміналов, які називають тілом або правою частиною продукції. Інтуїтивно призначення продукції - визначити один з можливих видів конструкції; якщо заголовний нетермінал представляє конструкцію, то тіло являє записуваний вигляд конструкції.

4. Один з нетермінальних символів, що вказується як стартовий або початковий [9].

Формально г**раматика** визначається як наступна четвірка компонентів (V_T, V_N, P, S).

Тут:

V_T – алфавіт термінальних символів або терміналів;

V_N – алфавіт нетермінальних символів або нетерміналів, $V_T \bigcap V_N = \varnothing$;

P – множина продукцій (або правил) вигляду $\alpha \to \beta$, α складається з одного або більше символів V, β – з нуля або більше символів V, де $V = V_T \bigcup V_N$;

S – стартовий символ (або аксіома) [10].

Приклад 1. Мова $\{x^n y^n \mid n > 0\}$ описується граматикою

$G_1 = (\{x, y\}, \{S\}, P, S)$.

Тут $P = \{S \to xSy, S \to xy\}$.

Приклад 2. Граматикою для мови $\{x^m y^n \mid m, n \geq 0\}$ є $G_2 = (\{x, y\}, \{S, B\}, P, S)$.

Тут набір продукцій P має вигляд

$S \to xS$, $S \to y$,

$S \to yB$, $B \to yB$,

$S \to x$, $B \to y$.

Оскільки порожній рядок також належить мові, у набір P також входить продукція $S \to \varepsilon$.

Рядок $xxyyy$ генерується в такий спосіб:

$S \Rightarrow xS \Rightarrow xxS \Rightarrow xxyB \Rightarrow xxyyB \Rightarrow xxyyy$ [11].

Нетермінали записуються словами в дужках <> або великими латинськими буквами. Термінали за необхідності часом беруться в апострофи. Як і в мові БНФ, замість продукцій вигляду $v \to w_1 w w_2$ і $v \to w_1 w_2$ записується продукція $v \to w_1 [w] w_2$,а замість продукцій $v \to w_1 u_1 w_2$ і $v \to w_1 u_2 w_2$ – продукція $v_1 \to w_1 (u_1 \mid u_2) w_2$ [9].

Приклад 3. Множину продукцій граматики $G_1 = (\{a, 1, 2\}, \{A, B, C, D\}, \{A \to BC, A \to BD, A \to B, B \to a, C \to 1, D \to 2\}, A)$ можна переписати у вигляді $\{A \to B[C \mid D], B \to a, C \to 1, D \to 2\}$.

Як бачимо, продукції граматики дуже схожі на БНФ (форма Бекуса-Наура) як за формою, так і за змістом – лише замість знака "::=" вживається "□ ". Проте в лівій частині їх продукцій може бути не поодинокий нетермінал, а цілий ланцюжок, у якому обов'язково є нетермінал. За рахунок такого узагальнення граматики виявляються більш потужним засобом задання мов, ніж системи БНФ, тобто існують мови, які задаються граматиками, але не задаються БНФ. Тепер опишемо спосіб, у який граматика $G = (V_T, V_N, P, S)$ задає мову [9].

1. На множині слів об'єднаного алфавіту $(V)^*$ означається ***відношення безпосередньої виводимості***, позначене знаком $\Rightarrow_G$ (або $\Rightarrow$, коли відомо, якою саме є G): $v \Rightarrow_G w$, якщо $v = x_1 u x_2$, $w = x_1 y x_2$, $u \to y \in P$.

При цьому кажуть, що *w безпосередньо виводиться з v* ***застосуванням продукції*** $u \to y$. Наприклад, у граматиці G_1 із прикладу **5** $BC \Rightarrow aC$ застосуванням продукції $B \to a, aC \Rightarrow a1$ застосуванням $C \to 1$.

2. На множині $(V)^*$ означається ***відношення виводимості***, позначене $\Rightarrow *_G$ (або $\Rightarrow *$, коли відомо, якою саме є G): $v \Rightarrow *w$, якщо $v = w$ або існує послідовність $w_1, w_2, ..., w_n$ слів, де $n \geq 1$,така, що $v \Rightarrow w_1, w_1 \Rightarrow w_2, ..., w_{n-1} \Rightarrow w_n, w_n = w$. Так, у граматиці $G_1 BC \Rightarrow *a1$,

оскільки $BC \Rightarrow aC,\ aC \Rightarrow a1$. Послідовність $v \Rightarrow w_1 \Rightarrow w_2 \Rightarrow ... \Rightarrow w_n$ називається ***виведенням*** w_n із v , а n – ***довжиною виведення***. Інколи довжину записують замість '*' таким чином: $v \Rightarrow^n w$, наприклад, $BC \Rightarrow^2 a1$.

3. Якщо $S \Rightarrow_G *w$, то послідовність $S \Rightarrow ... \Rightarrow w$ називається ***виведенням слова w у граматиці*** G , а слово w – ***вивідним***. Так, слова .. вивідні в граматиці *прикладу 5*.

4. Вивідні слова в алфавіті X називаються ***породжуваними***, а множина їх усіх – ***мовою, що задається*** (***породжується***) ***граматикою*** $G: L(G) = \{w \mid w \in X^* \text{ та } S \Rightarrow^* w\}$ [10].

Приклад 4. Граматика G_1 із прикладу 3 задає мову $\{a, a1, a2\}$.

Приклад 5. Граматика

$G_2 = (\{a,...,z,0,...,9\}, \{I, L, D\}, \{I \to L \mid IL \mid ID, L \to a \mid ... \mid z, D \to 0 \mid ... \mid 9\}, I)$ породжує множину ідентифікаторів.

Приклад 6. Граматика $G_3 = (\{(,)\}, \{S\}, \{S \to \varepsilon \mid (S)\}, S)$ **)** задає множину "вкладених дужок" $\{(^n)^n \mid n \geq 0\}$.

Приклад 7. Граматика

$G_4 = (\{a,b,c\}, \{S, A, B, C\}, \{S \to aSBC \mid abC, CB \to BC, bB \to bb,$

$bC \to bc,\ cC \to cc\},\ S)$ визначає мову $\{a^n b^n c^n \mid n \geq 1\}$.

Граматики називаються ***еквівалентними***, якщо задають ту саму мову. Наприклад, граматика $(\{a,1,2\}, \{A\}, \{A \to a[1 \mid 2]\}, A)$ еквівалентна граматиці *з прикладу 5*, граматика

$(\{a,...,z,0,...,9\}, \{I, C\}, \{I \to (a \mid ... \mid z)C, C \to \varepsilon \mid C(a \mid ... \mid z \mid 0 \mid ... \mid 9)\}, I)$ – граматиці *з прикладу 7*.

Кожен з рядків, що фігурують у породженні, називається **сентенціальною формою**, а останній рядок, що складається з терміналів, – **сентенцією** (пропозицією) мови.

При цьому

$\Rightarrow$ означає один крок,

$\overset{*}{\Rightarrow}$ нуль або більше кроків,

$\overset{+}{\Rightarrow}$ один або більше кроків.

Запис $\begin{cases} B \to yB, \\ B \to y \end{cases}$ можна записувати у більш стислому вигляді: $B \to yB \mid y$.

Ту ж саму мову можна згенерувати багатьма граматиками.

Наше визначення граматики допускає граматики і більш загальних типів, ніж були у прикладах.

Наприклад: $G_3 = (\{a\}, \{S, N, Q, R\}, P, S)$, де P містить продукції:

$S \to QNQ$,

$Q \to QR$,

$RQ \to NNQ$,

...

Тут N переходить у R, якщо N стоїть після Q, а R переходить у NN, якщо воно стоїть перед Q.

У даному прикладі продукції є контекстозалежними**.** Контекстнонезалежні продукції мають порожній контекст [11].

3.2. Алгоритмізація задачі за темою роботи

Початковий крок. На панелі виводиться інформація:

- тема;
- назва дистанційного курсу;
- ПІБ автора;
- поточний рік;
- кнопка для переходу до тестування.
- кнопка для переходу до прикладів.

Алгоритм тестування (правильна відповідь підкреслена):

1 крок. На панелі виводиться питання та варіанти відповіді:

«Граматика має чотири компоненти:

- Безліч термінальних символів, іноді іменованих токенів.
- Безліч нетермінальних символів, іноді іменованих токенів.
- Безліч терміналів, які іноді називають синтаксичними змінними.

- Безліч нетерміналів, які іноді називають синтаксичними змінними.
- Безліч продукцій, кожна з яких складається з термінала, який називають заголовком або лівою частиною продукції.
- Безліч продукцій, кожна з яких складається з нетермінала, який називають заголовком або лівою частиною продукції.
- Один з нетермінальних символів, що вказується як стартовий або початковий.
- Один з термінальних символів, що вказується як стартовий або початковий».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

2 крок. На панелі виводиться питання та варіанти відповіді:

«Формально граматика визначається як наступна четвірка компонентів (___, ___, ___, ___). Встановіть послідовність компонентів.

Тут:

1. V_T – алфавіт термінальних символів або терміналів;
2. V_N – алфавіт нетермінальних символів або нетерміналів, $V_T \cap V_N = \varnothing$;
3. P – множина продукцій (або правил) вигляду $\alpha \to \beta$, α складається з одного або більше символів V, β – з нуля або більше символів V, де $V = V_T \cup V_N$;
4. S – стартовий символ (або аксіома)».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

3 крок. На панелі виводиться питання та варіанти відповіді:

«Нетермінали записуються:

- словами в дужках [].
- словами в дужках <>.

- великими латинськими буквами.
- латинськими буквами»

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

4 крок. На панелі виводиться питання та варіанти відповіді:

«Термінали за необхідності часом беруться в:

- апострофи.
- в дужки <>.
- в дужки ()».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

5 крок. На панелі виводиться питання та варіанти відповіді:

«Як і в мові БНФ, замість продукцій вигляду $v \to w_1 w w_2$ і $v \to w_1 w_2$ записується продукція:

- $v \to w_1[w]w_2$.
- $v \to [w]w_1 w_2$.
- $v \to w_1 w_2$».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

6 крок. На панелі виводиться питання та варіанти відповіді:

«Як і в мові БНФ, замість продукцій вигляду $v \to w_1 u_1 w_2$ і $v \to w_1 u_2 w_2$ записується продукція:

- $v_1 \to w_1(u_1 u_2)w_2$.
- $v_1 \to w_1(u_1 \mid u_2)w_2$.
- $v_1 \to w_1[u_1 \mid u_2]w_2$».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

7 крок. На панелі виводиться питання та варіанти відповіді:

«Тепер опишемо спосіб, у який граматика $G=(V_T,V_N,P,S)$ задає мову.

На множині слів об'єднаного алфавіту $(V)^*$ означається ***відношення***, позначене знаком $\Rightarrow_G$ (або $\Rightarrow$, коли відомо, якою саме є G): $v \Rightarrow_G w$, якщо $v = x_1 u x_2$, $w = x_1 y x_2$, $u \to y \in P$, як:

- відношення безпосередньої виводимості.
- відношення виводимості.
- виведенням слова w у граматиці G».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

8 крок. На панелі виводиться питання та варіанти відповіді:

«На множині слів об'єднаного алфавіту $(V)^*$ означається ***відношення безпосередньої виводимості***, позначене знаком $\Rightarrow_G$ (або $\Rightarrow$, коли відомо, якою саме є G): $v \Rightarrow_G w$, якщо $v = x_1 u x_2$, $w = x_1 y x_2$, $u \to y \in P$.

При цьому кажуть, що *w безпосередньо виводиться з v*:

- застосуванням продукції $y \to u$.
- застосуванням продукції $u \to y$.
- виведенням слова w у граматиці G».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

9 крок. На панелі виводиться питання та варіанти відповіді:

«На множині $(V)^*$ означається ***відношення***, позначене $\Rightarrow^*_G$ (або $\Rightarrow^*$, коли відомо, якою саме є G): $v \Rightarrow^* w$, якщо $v = w$ або існує послідовність $w_1, w_2, ..., w_n$ слів, де $n \geq 1$,така, що $v \Rightarrow w_1, w_1 \Rightarrow w_2, ..., w_{n-1} \Rightarrow w_n, w_n = w$, як:

- відношення безпосередньої виводимості.
- відношення виводимості.

- виведенням слова w у граматиці G».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

10 крок. На панелі виводиться питання та варіанти відповіді:

«На множині $(V)^*$ означається ***відношення виводимості***, позначене $\Rightarrow^*_G$ (або $\Rightarrow^*$, коли відомо, якою саме є G): $v \Rightarrow^* w$, якщо $v = w$ або існує послідовність $w_1, w_2, ..., w_n$ слів, де $n \geq 1$,така, що $v \Rightarrow w_1, w_1 \Rightarrow w_2, ..., w_{n-1} \Rightarrow w_n, w_n = w$.

Послідовність $v \Rightarrow w_1 \Rightarrow w_2 \Rightarrow ... \Rightarrow w_n$ називається:

- довжиною виведення.
- виведенням v із w_n.
- виведенням w_n із v».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

11 крок. На панелі виводиться питання та варіанти відповіді:

«Послідовність $v \Rightarrow w_1 \Rightarrow w_2 \Rightarrow ... \Rightarrow w_n$ називається ***виведенням*** w_n із v, а n:

- довжиною виведення.
- виведенням v із w_n.
- виведенням w_n із v».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

12 крок. На панелі виводиться питання та варіанти відповіді:

«Якщо $S \Rightarrow_G {}^* w$, то послідовність $S \Rightarrow ... \Rightarrow w$ називається:

- відношення безпосередньої виводимості.
- відношення виводимості.
- виведенням слова w у граматиці G».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

13 крок. На панелі виводиться питання та варіанти відповіді:

«Якщо $S \Rightarrow_G {}^*w$, то послідовність $S \Rightarrow ... \Rightarrow w$ називається ***виведенням слова*** w ***у граматиці*** G, а слово w:

- вивідним.
- невивідним.
- породжуваним».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

14 крок. На панелі виводиться питання та варіанти відповіді:

«Вивідні слова в алфавіті X називаються:

- невивідними.
- еквівалентними.
- породжуваними».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

15 крок. На панелі виводиться питання та варіанти відповіді:

«Вивідні слова в алфавіті X називаються ***породжуваними***, а множина їх усіх:

- мовою, що задається (породжується) граматикою $G : L(G) = \{w \mid w \in X^* \text{ та } S \Rightarrow {}^*w\}$.
- мовою, що не задається (породжується) граматикою $G : L(G) = \{w \mid w \in X^* \text{ та } S \Rightarrow {}^*w\}$.
- граматикою $G : L(G) = \{w \mid w \in X^* \text{ та } S \Rightarrow {}^*w\}$».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

16 крок. На панелі виводиться питання та варіанти відповіді:

«Граматики називаються ***еквівалентними***, якщо:

- не задають ту саму мову.
- задають ту саму мову.
- вони однакові».

При неправильно обраних варіантах вказується помилка. При правильному – перехід далі на крок.

Кінцевий крок. На панелі виводиться інформація:

- повідомлення про завершення тестування;
- перелік кроків;
- кількість допущених помилок;
- кнопка для повторного тестування.

Якщо натиснути кнопку, то відображається перший крок.

Алгоритм виконання прикладів (правильна відповідь вказується в дужках, правильна відповідь підкреслена):

1 крок. На панелі виводиться приклад та поле(-я) для відповіді:

«Мова $\{x^n y^n \mid n > 0\}$ описується граматикою

$G_1 = (\{x, y\}, \{S\}, P, S)$.

Вкажіть P:

- $P = \{\_\_\_, \_\_\_\}$» ($P = \{S \to xSy, S \to xy\}$)

При неправильній відповіді вказується помилка. При правильному – перехід далі на крок.

2 крок. На панелі виводиться приклад та поле(-я) для відповіді:

«Граматикою для мови $\{x^m y^n \mid m, n \geq 0\}$ є $G_2 = (\{x, y\}, \{S, B\}, P, S)$.

Тут набір продукцій *P* має вигляд

$S \to xS$, $S \to y$,

$S \to yB$, $B \to yB$,

$S \to x$, $B \to y$.

$S \to \varepsilon$

Вкажіть порядок як генерується *xxyyy*:

- $S \Rightarrow \_\_\_ \Rightarrow \_\_\_ \Rightarrow \_\_\_ \Rightarrow \_\_\_ \Rightarrow xxyyy$»

  ($S \Rightarrow xS \Rightarrow xxS \Rightarrow xxyB \Rightarrow xxyyB \Rightarrow xxyyy$)

При неправильній відповіді вказується помилка. При правильному – перехід далі на крок.

3 крок. На панелі виводиться приклад та поле(-я) для відповіді:

«Множину продукцій граматики $G_1 = (\{a,1,2\},\{A,B,C,D\},$ $\{A \to BC, A \to BD, A \to B, B \to a, C \to 1, D \to 2\}, A)$ можна переписати у вигляді:

- $\{A \to \_\_\_\_, B \to \_\_\_, C \to \_\_\_, D \to \_\_\_\}$»

  ($\{A \to B[C \mid D], B \to a, C \to 1, D \to 2\}$)

При неправильній відповіді вказується помилка. При правильному – перехід далі на крок.

4 крок. На панелі виводиться приклад та поле(-я) для відповіді:

«Граматика $G_1 = (\{a,1,2\},\{A,B,C,D\},$ задає мову:

- $\{\_\_\_, \_\_\_, \_\_\_\}$»

  ($\{a, a1, a2\}$)

При неправильній відповіді вказується помилка. При правильному – перехід далі на крок.

5 крок. На панелі виводиться приклад та варіанти відповіді:

«Яка граматика породжує множину ідентифікаторів?

- $G = (\{(,)\},\{S\},\{S \to \varepsilon \mid (S)\}, S)$
- $G = (\{a,b,c\},\{S,A,B,C\},\{S \to aSBC \mid abC, CB \to BC, bB \to bb,$ $bC \to bc, cC \to cc\}, S)$
- $G = (\{a,...,z,0,...,9\},\{I,L,D\},\{I \to L \mid IL \mid ID, L \to a \mid ... \mid z,$ $D \to 0 \mid ... \mid 9\}, I)$»

При неправильній відповіді вказується помилка. При правильному – перехід далі на крок.

6 крок. На панелі виводиться приклад та варіанти відповіді:

«Яка граматика задає множину "вкладених дужок" $\{(^n)^n \mid n \geq 0\}$?

- $G = (\{(,)\},\{S\},\{S \to \varepsilon \mid (S)\}, S)$
- $G = (\{a,b,c\},\{S,A,B,C\},\{S \to aSBC \mid abC, CB \to BC, bB \to bb, bC \to bc,\ cC \to cc\},\ S)$
- $G = (\{a,...,z,0,...,9\},\{I,L,D\},\{I \to L \mid IL \mid ID, L \to a \mid ... \mid z, D \to 0 \mid ... \mid 9\}, I)$ »

При неправильній відповіді вказується помилка. При правильному – перехід далі на крок.

7 крок. На панелі виводиться приклад та варіанти відповіді:

«Яка граматика визначає мову $\{a^n b^n c^n \mid n \geq 1\}$?

- $G = (\{(,)\},\{S\},\{S \to \varepsilon \mid (S)\}, S)$
- $G = (\{a,b,c\},\{S,A,B,C\},\{S \to aSBC \mid abC, CB \to BC, bB \to bb, bC \to bc,\ cC \to cc\},\ S)$
- $G = (\{a,...,z,0,...,9\},\{I,L,D\},\{I \to L \mid IL \mid ID, L \to a \mid ... \mid z, D \to 0 \mid ... \mid 9\}, I)$ »

При неправильній відповіді вказується помилка. При правильному – перехід далі на крок.

Кінцевий крок. На панелі виводиться інформація:

- повідомлення про виконання прикладів;
- перелік кроків;
- кількість допущених помилок;
- кнопка для повторного проходження.

3.3. Розробка блок-схеми, яка підлягає програмуванню

На рисунку 3.1 зображена блок-схема роботи тренажеру, на рисунку 3.2 – блок-схема процесу при виборі відповіді, на рисунку 3.3 – блок-схема процесу при введенні відповіді.

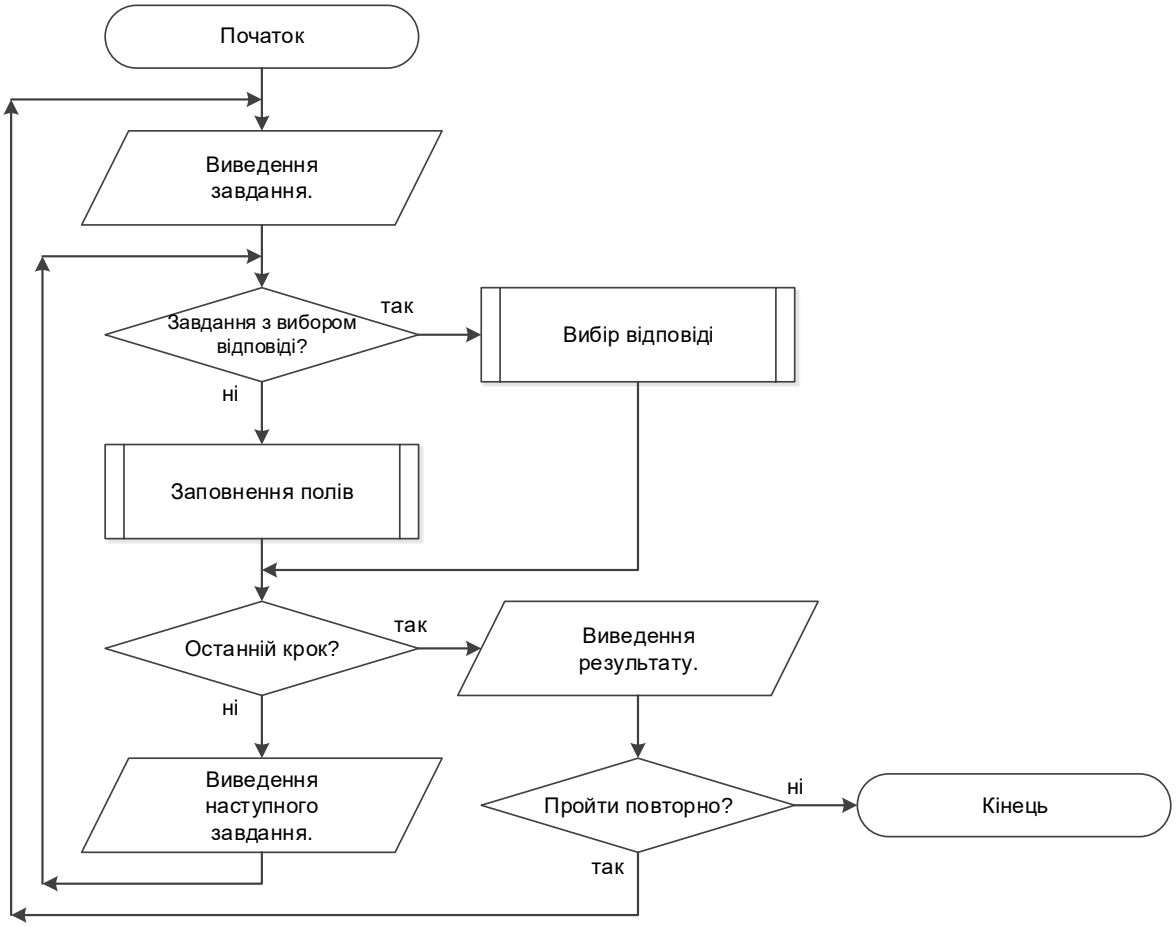

Рисунок 3.1 – Блок-схема роботи програми

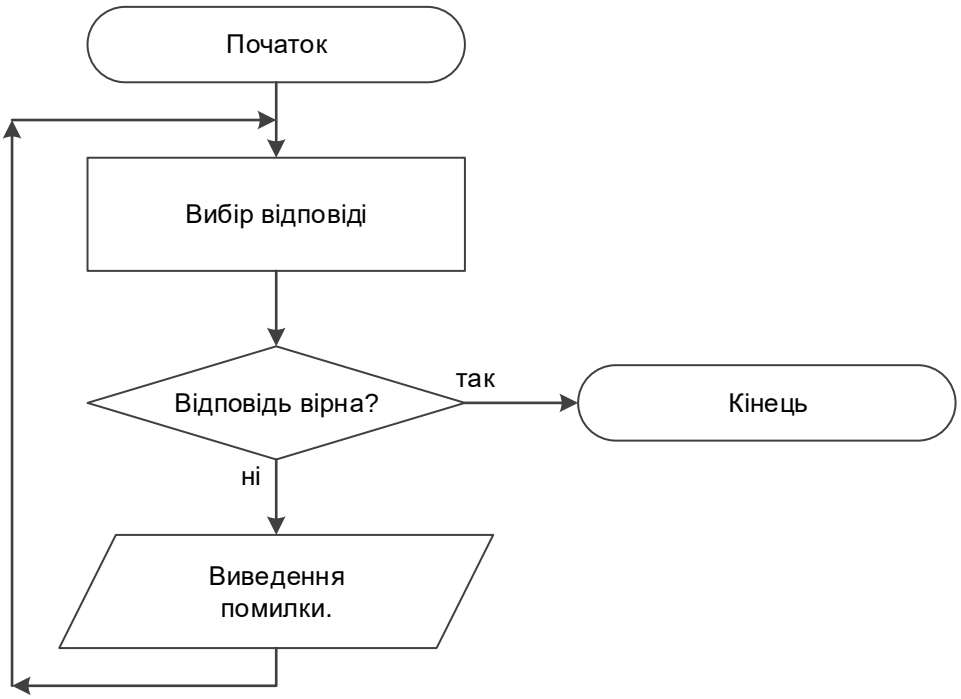

Рисунок 3.2 – Блок-схема процесу при виборі відповіді

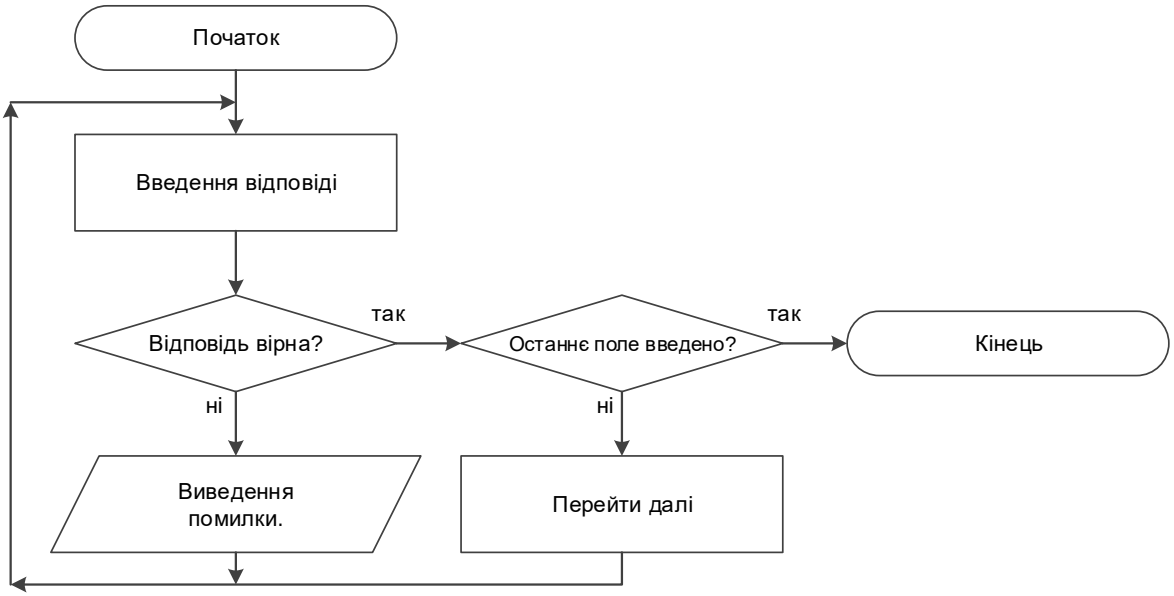

Рисунок 3.3 – Блок-схема процесу при введенні відповіді

4. ПРАКТИЧНА ЧАСТИНА

4.1. Обґрунтування вибору програмних засобів для реалізації завдання роботи

Значну роль у виборі програмного середовища відіграє його функціонал. Для того, щоб виступати ефективним засобом навчання, функціонал програмного засобу повинен відповідати змісту навчання. Проте трапляються випадки, коли більший функціонал програмного середовища спонукатиме по подальшого самостійного вивчення програмування.

Наприклад, повна версія Microsoft Visual Studio Standard Edition платна, причому її ціна досить висока. Безкоштовна версія Microsoft Visual Studio Express Edition має деякі обмеження стосовно функціоналу, однак дані обмеження не відіграють великого значення під час навчання, оскільки стосуються лише тих особливостей, які у школі, зазвичай, не вивчаються.

Обмежений функціонал програмного середовища SharpDevelop майже не впливає на його дидактичні можливості, адже стає значущим лише під час розроблення складних програм, які взаємодіють з базами даних тощо. Однак функціональних можливостей SharpDevelop версії, починаючи з 4.0 і вище, цілком достатньо для навчання програмування мовою C# і створення програм з досить складним алгоритмом. Інший приклад – Eclipse і NetBeans. У середовищі Eclipse можна розробляти повнофункціональні програми мовою Java і перетворити програму в EXE-файл, але неможливо розмістити компоненти на екранній формі у візуальному режимі. У середовищі NetBeans є візуальний режим роботи з екранною формою, однак відсутня функція перетворення програми на EXE-файл.

Під час вибору програмного середовища слід враховувати також особливості інтерфейсу. До таких особливостей слід віднести:

- вид інтерфейсу (графічний або командний рядок);
- раціональне компонування інтерфейсу. Доступ до основних функцій і до основних компонентів повинен бути якомога простішим;
- мова інтерфейсу значно полегшує процес засвоєння прийомів роботи у програмному середовищі;
- наявність підказки значно спрощує процес виправлення помилок;
- наявність документації на програмне середовище. Цей чинник впливає на швидкість засвоєння програмного середовища вчителем;
- наявність відповідного методичного забезпечення залежить від поширення програмного середовища у навчальних закладах.

Однією з характеристик програмного середовища є можливість роботи з візуальними компонентами, яка дозволяє розробляти програми у візуальному режимі. Використання консольного введення-виведення ускладнює процес засвоєння учнями основ роботи у середовищі програмування і значно сповільнює процес розробки програм. Це негативно впливає на рівень інтересу учнів до програмування.

Навпаки, графічний інтерфейс сприяє активізації роботи учнів. Учні змінюють положення компонентів на формі і відразу бачать внесені зміни. Це сприяє підтриманню інтересу учнів до програмування на необхідному рівні, мотивує їх на подальшу роботу [12].

Мова С і її подальший розвиток (С++) – це потужна мова програмування, яка підтримує процедурну й об’єктно-орієнтовану парадигми програмування. Мови С/С++ широко використовуються для створення як прикладного, так і системного програмного забезпечення.

Мова С# набула значного поширення останнім часом. Мова С# розроблена і вдосконалюється фірмою Microsoft. С# має синтаксис,

подібний до синтаксису C++. Структура програми аналогічна до структури програми мовою C++.

На основі аналізу встановлено, що найбільш поширеними є такі середовища програмування мовою C#: Microsoft Visual Studio, SharpDevelop, MonoDevelop та інші.

Дане середовище містить більшість потрібних файлів бібліотек, які можна підключати в міру необхідності. Microsoft Visual Studio є професійним середовищем, у ньому можна створювати повноцінні, повнофункціональні програмні засоби.

Microsoft Visual Studio – одне з кількох середовищ розробки програм мовою C#. Стосовно написання програм на C#, то Microsoft Visual Studio має більші переваги перед іншими, оскільки повнофункціональним є лише воно [12].

Мова Java на даний час є однією з найбільш поширених мов програмування. Її було розроблено фірмою Sun Microsystems на початку 90-х років минулого століття, і дана мова продовжує розвиватися і вдосконалюватися.

Найбільш поширеними середовищами програмування мовою Java є IntelliJ IDEA, Eclipse, NetBeans, JBuilder. Для роботи кожного з названих середовищ необхідне встановлення платформи JDK і пакета JRE, тому ми не будемо розглядати дану особливість середовища програмування (вона характерна для всіх засобів розроблення програм мовою Java).

Розглянемо детальніше середовища програмування мовою Java.

Наступне середовище – NetBeans – належить до вільно-поширюваного програмного забезпечення. Інтерфейс його англомовний, підказка також англомовна. У середовищі NetBeans доступний візуальний режим створення форм і розміщення на них компонентів. Для навчання програміста-початківця це дуже важливо, оскільки він має можливість бачити зовнішній вигляд форми програми у процесі розроблення.

Системні вимоги NetBeans 8.0 допускають встановлення середовища на більшість комп’ютерів і комфортну роботу у ньому. Системні вимоги:

– процесор Intel Pentium III 800 МГц або еквівалент;
– оперативна пам’ять 512 МБ;
– обсяг вільного місця на диску 750 МБ [13].

4.2. Опис процесу програмної реалізації

Для переходу від одної панелі до іншої було використано CardLayout, а для виведення помилки в діалоговому вікні JOptionPane. Тому спочатку підключено ці бібліотеки.

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

Також створено змінні step і error для визначення поточного кроку та підрахунку помилок.

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

Було розроблено наступну структуру програми (рис. 4.1):

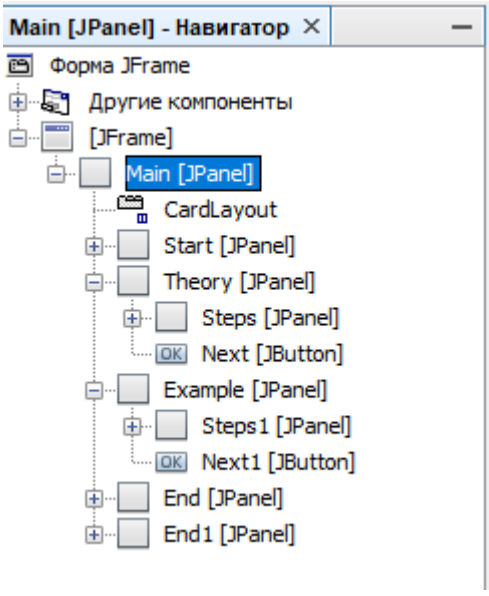

Рисунок 4.1 – Структура програми

Для кожної частини програми створені окремі кроки (рис. 4.2).

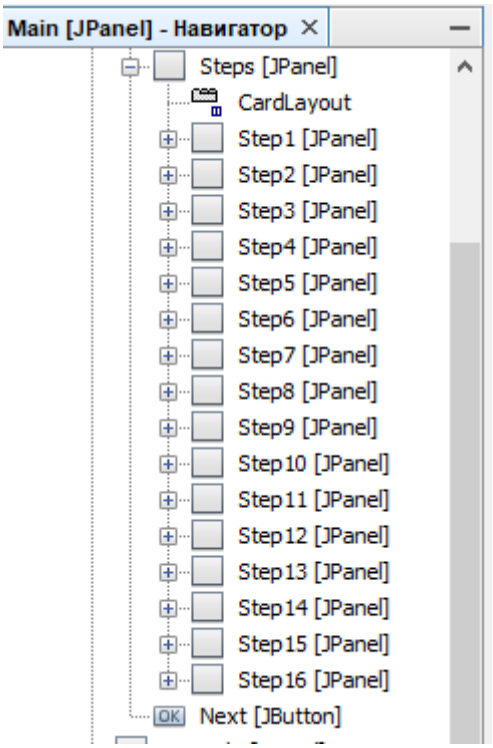

Рисунок 4.2 – Окремі кроки

Після цього було розроблено допоміжні функції. Так, void nextMain(String t) здійснює перехід до наступної основної панелі.

```
private void nextMain(String t) {
    card = (CardLayout) Main.getLayout();
    card.show(Main, t);
}
```

Кроки переключаються завдяки void nextTheory(int i) в тестуванні і void nextExample(int i) при виконанні прикладів.

```
private void nextTheory(int i) {
    card = (CardLayout) Steps.getLayout();
    card.show(Steps, "step"+i);
}

private void nextExample(int i) {
```

```
        card = (CardLayout) Steps1.getLayout();
        card.show(Steps1, "examp"+i);
    }
```

Окремо виділено функцію void showError() для виведення вікна з помилкою.

```
    private void showError() {
        JOptionPane.showMessageDialog(Main,
"<html>&ndsp;<b>Увага!</b><br> Надана відповідь неправильна, перевірте її.<br>", "Помилка", JOptionPane.ERROR_MESSAGE);
    }
```

Всі інші дії виконуються завдяки подіям, що спрацьовують при натисканні відповідної кнопки. При переході до тестування спрацьовує jButton1ActionPerformed, що відображає перший крок алгоритму тестування.

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
        nextMain("theory");
        nextTheory(1);
    }
```

Подія jButton2ActionPerformed відповідає за перехід до виконання першого прикладу.

```
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{
        nextMain("example");
        nextExample(1);
    }
```

Перевірка відповіді відбувається за допомогою NextActionPerformed та Next1ActionPerformed для тестування і виконання завдань відповідно. Вони схожі за своєю роботою (див. Додаток А).

Також після проходження тренажеру виводиться результат, де можна знову вибрати наступну дію: почати спочатку або перейти до іншого блоку завдань. Всі ці події працюють аналогічно як при першому виборі на стартовій панелі.

```
        private void jButton3ActionPerformed(java.awt.event.ActionEvent evt)
{
            step=1;
            error=0;
            nextMain("theory");
            nextTheory(1);
        }

        private void jButton4ActionPerformed(java.awt.event.ActionEvent evt)
{
            step=1;
            error=0;
            nextMain("example");
            nextExample(1);
        }
```

4.3. Опис роботи програми

На стартовій панелі виводиться тема, назва дистанційного курсу, ПІБ автора, поточний рік, кнопка для переходу до тестування, кнопка для переходу до прикладів (рис. 4.3).

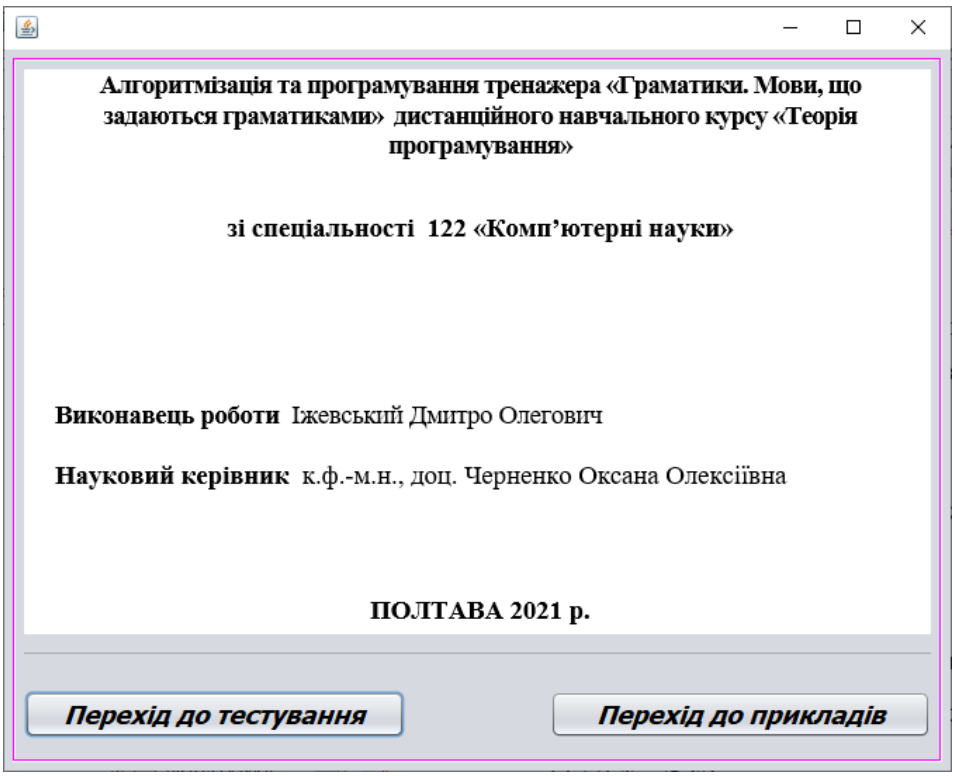

Рисунок 4.3 – Стартова панель

Щоб перейти до тестування необхідно натиснути кнопку «Перехід до тестування». Після цього відображається перший крок тестування (рис. 4.4).

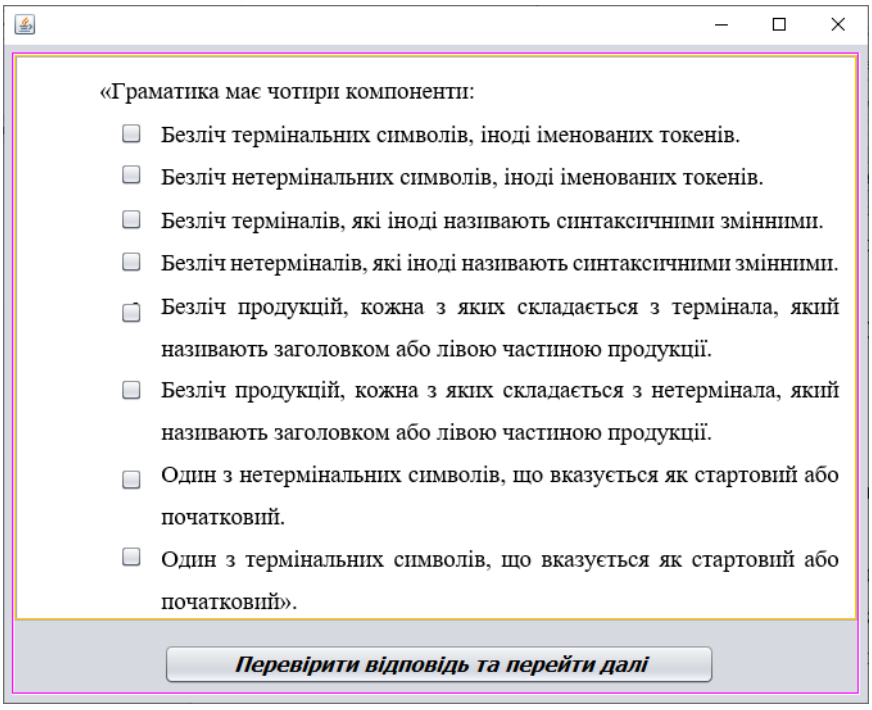

Рисунок 4.4 – Перший крок тестування

В даному випадку потрібно вибрати відповідь, але це може бути не лише один з варіантів, а й декілька (рис. 4.5). Тому слід детально вибирати. Якщо відповідь надано неправильно, то виведеться помилка (рис. 4.6).

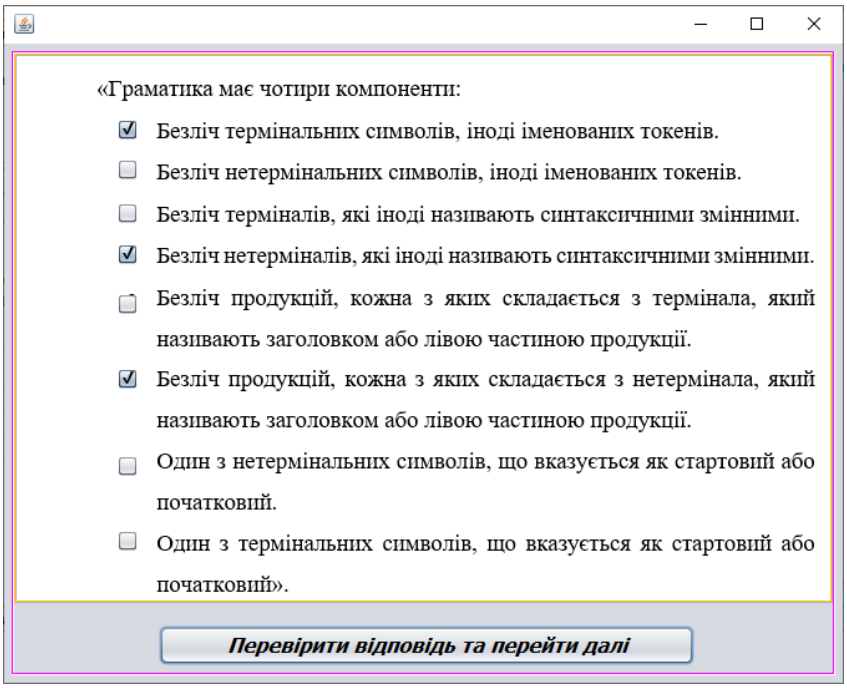

Рисунок 4.5 – Вибір відповіді

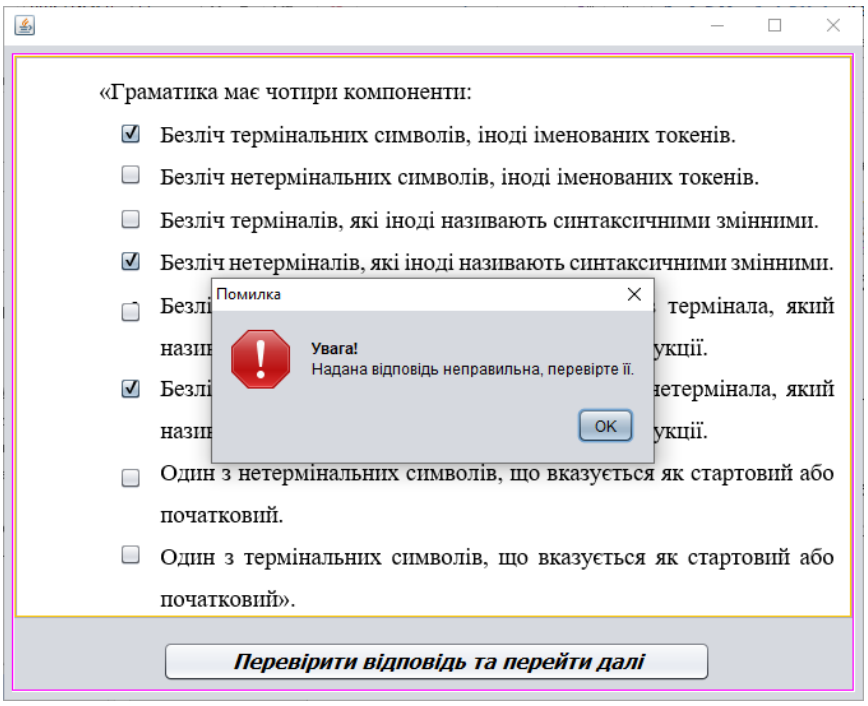

Рисунок 4.6 – Помилкам

При правильній відповіді відбудеться перехід до наступного тесту (рис. 4.7). Тут потрібно встановити послідовність (рис. 4.8).

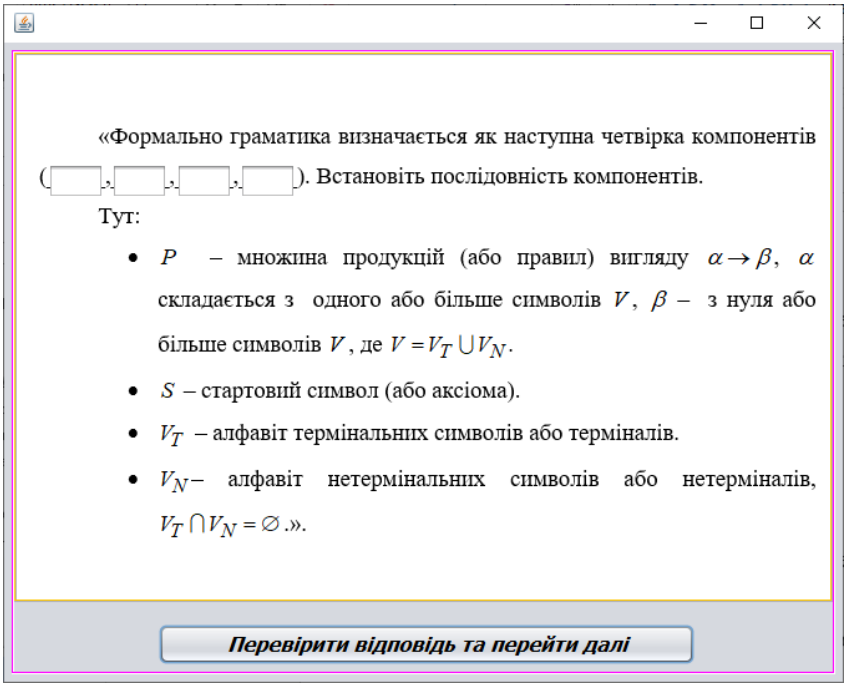

Рисунок 4.7 – Другий крок тестування

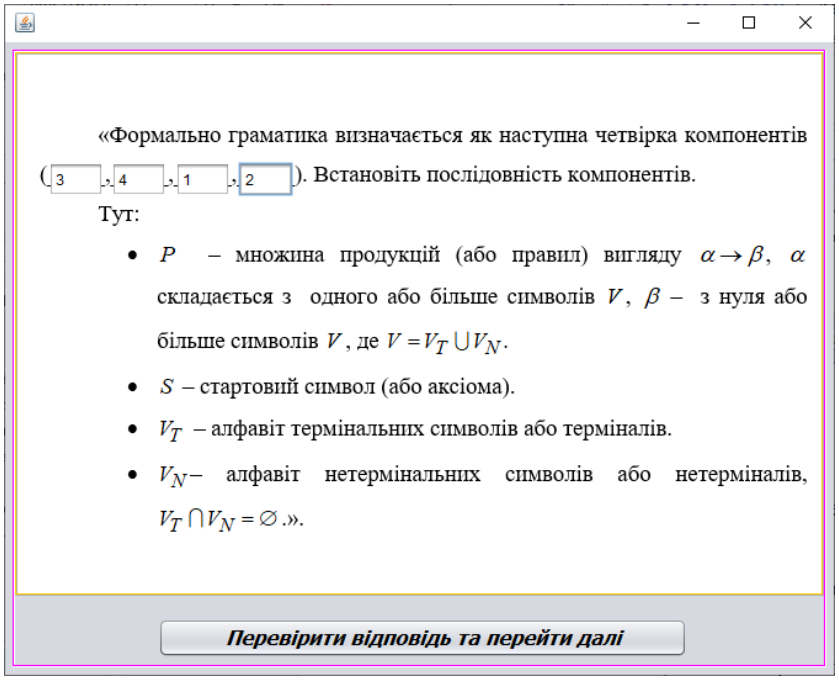

Рисунок 4.8 – Встановлення послідовності

Якщо все правильно, то перехід до наступного тесту (рис. 4.9) і т.д.

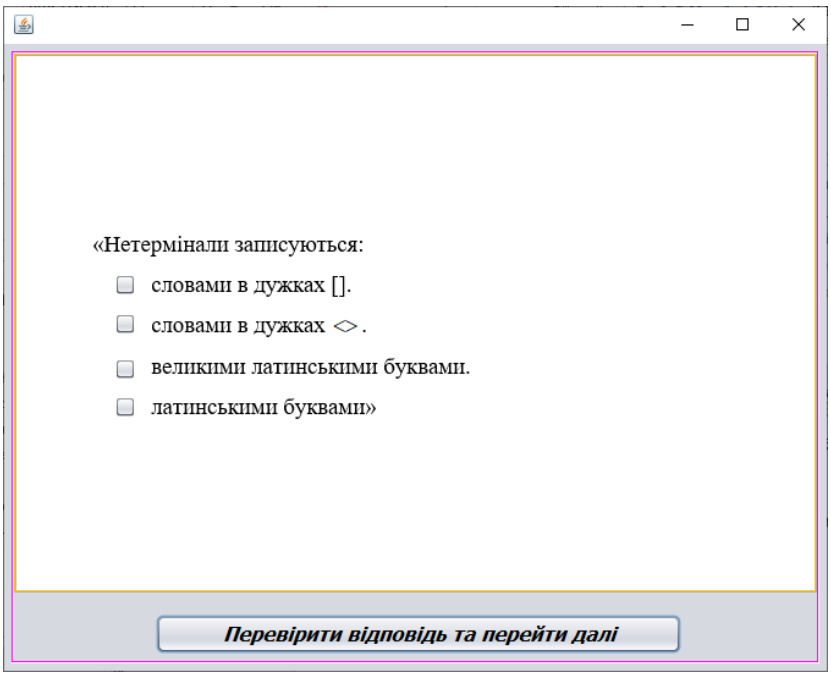

Рисунок 4.9 – Третій крок тестування

Після завершення тестування буде виведено результат, де буде вказано кількість пройдених кроків і помилок (рис. 4.10). Знову можна вибрати наступну дію.

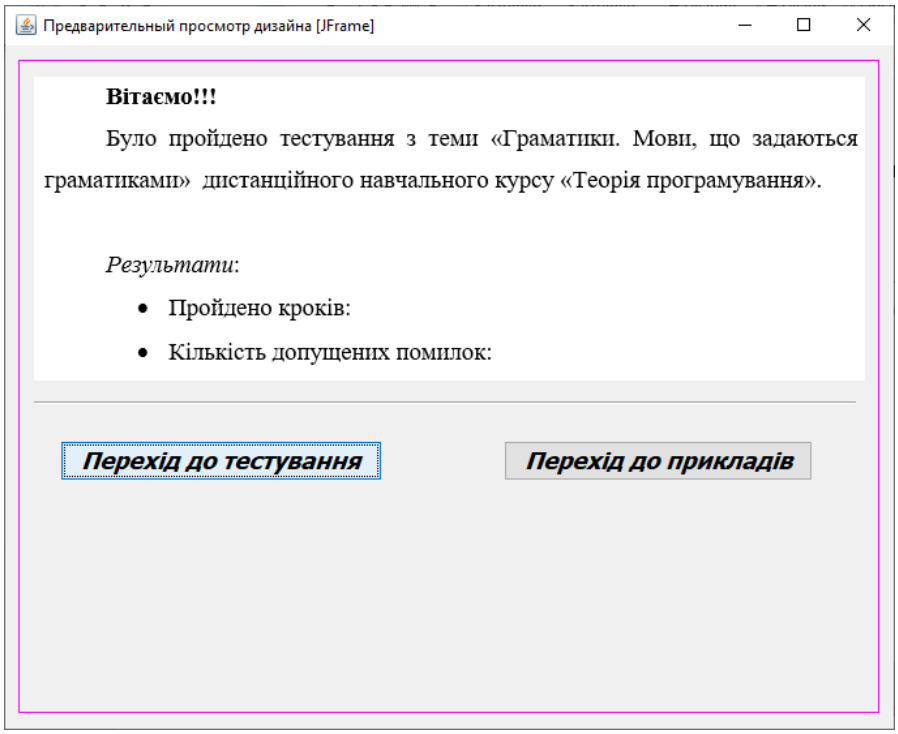

Рисунок 4.10 – Завершення тестування

Щоб перейти до прикладів необхідно натиснути кнопку «Перехід до прикладів». Після цього відображається перший приклад (рис. 4.11).

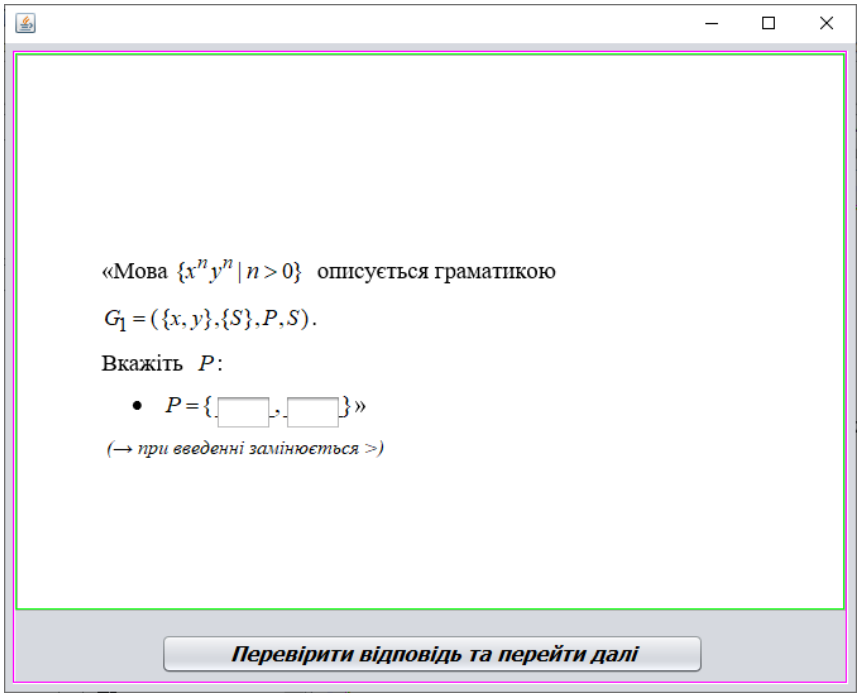

Рисунок 4.11 – Перший приклад

В даному випадку потрібно ввести відповідь (рис. 4.12). Якщо відповідь надано неправильно, то виведеться помилка.

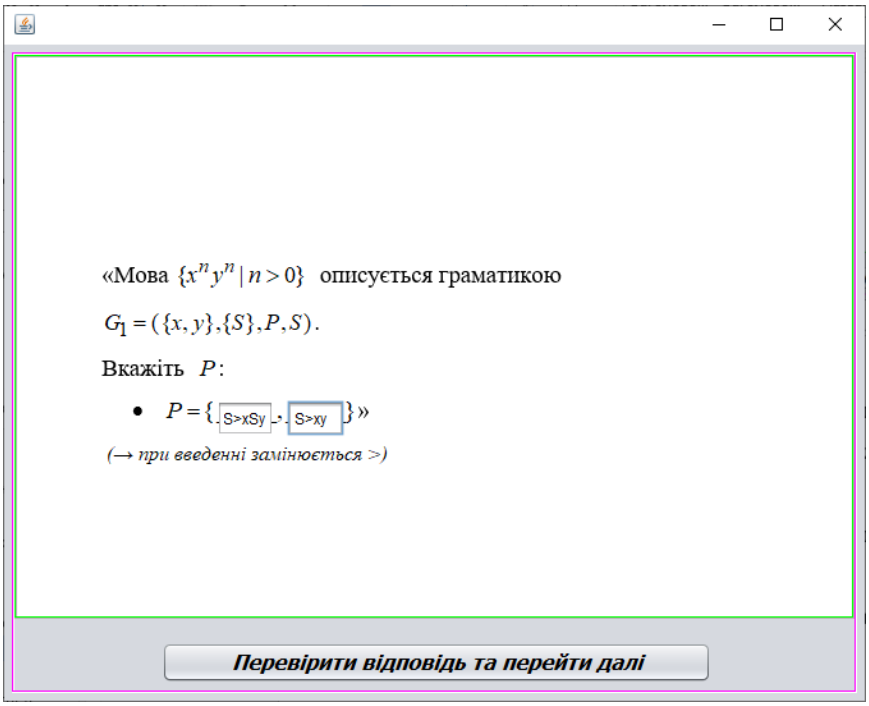

Рисунок 4.12 – Введення відповіді

При правильній відповіді відображається наступний приклад (рис. 4.13) і т.д.

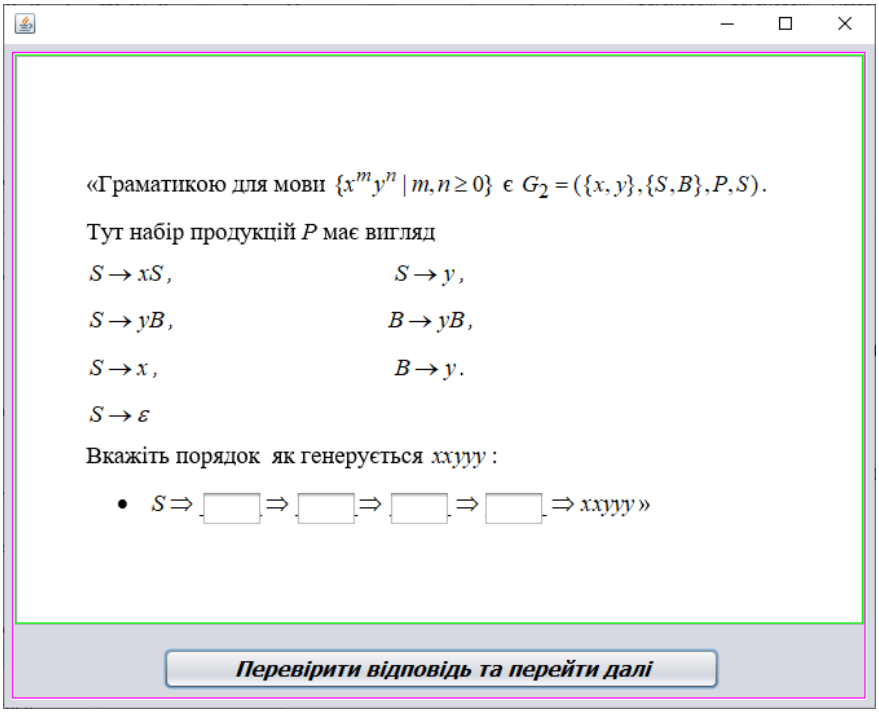

Рисунок 4.13 – Другий приклад

В кінці буде виведено результат, де буде знову вказано кількість пройдених кроків і помилок (рис. 4.14). Знову можна вибрати наступну дію.

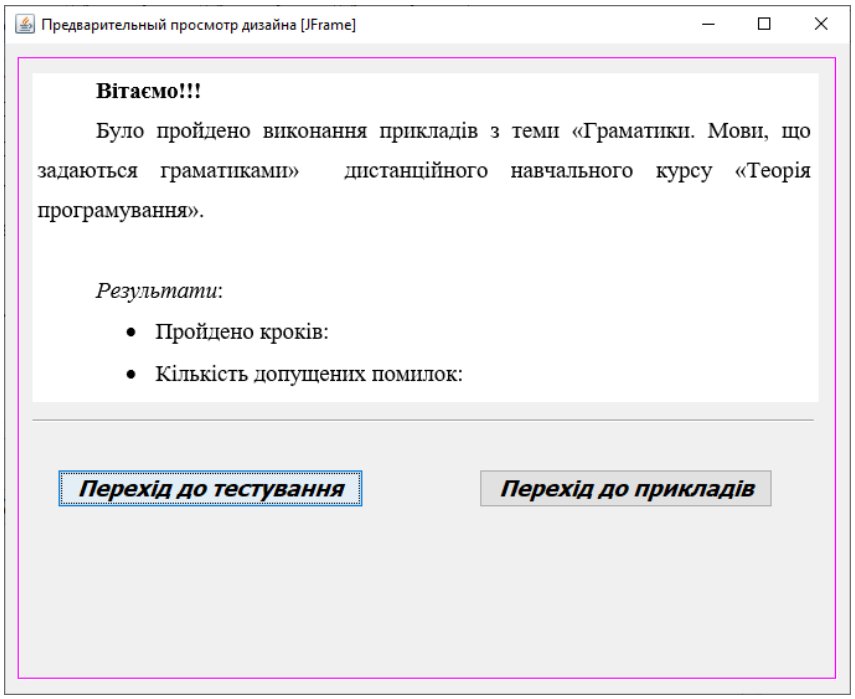

Рисунок 4.14 – Завершення виконання прикладів

ВИСНОВКИ

Метою виконаної роботи є алгоритмізація та програмування тренажеру «Граматики. Мови, що задаються граматиками» дистанційного навчального курсу «Теорія програмування».

В роботі описано такі пункти:

- Постановка задачі;
- Типи комп'ютерних тренажерів;
- Системи, середовища програмування, середовища для розробки програмного забезпечення;
- Актуальність теми;
- Огляд матеріалу з теми;
- Алгоритм робити тренажеру;
- Блок-схема роботи тренажеру;
- Обґрунтування вибору програмних засобів;
- Опис процесу розробки;
- Опис роботи програми.

Незважаючи на те, що технологічна основа навчального процесу у вищий школі, у тому числі і сучасні інформаційні технології, швидко розвиваються, до цього часу не існує єдиної системи навчального інформаційно-програмного середовища. У зв’язку з цим, у більшості випадків комп’ютерне забезпечення дисциплін розробляється автономно з орієнтацією на конкретну дисципліну та конкретний навчальний заклад. Як правило, це сукупність програмного, текстового, методичного, тестового, аудіо та відео супроводу, що практично дублює інформацію з розділів підручників та попередніх методичних розробок.

Застосування сучасних комп’ютерних і телекомунікаційних технологій в навчальному процесі не тільки створює умови для більш ефективної самостійної роботи студентів, сприяє індивідуалізації процесу підготовки фахівців, а і суттєво змінює форми і зміст комунікацій між

викладачем і студентом. За допомогою комп'ютерних технологій, незважаючи на незмінні тенденції до зменшення аудиторних годин, прямий і зворотній зв'язок «викладач-студент» стає більш інтенсивним і активним.

Було розглянуто приклади і використано при реалізації тренажеру.

Приклад 1. Мова $\{x^n y^n \mid n > 0\}$ описується граматикою

$G_1 = (\{x, y\}, \{S\}, P, S)$.

Вказати P.

Приклад 2. Граматикою для мови $\{x^m y^n \mid m, n \geq 0\}$ є $G_2 = (\{x, y\}, \{S, B\}, P, S)$.

Тут набір продукцій P має вигляд

$S \to xS$, $S \to y$,

$S \to yB$, $B \to yB$,

$S \to x$, $B \to y$.

Оскільки порожній рядок також належить мові, у набір P також входить продукція $S \to \varepsilon$.

Як генерується рядок $xxyyy$?

Приклад 3. Як можна переписати множину продукцій граматики $G_1 = (\{a,1,2\}, \{A,B,C,D\}, \{A \to BC, A \to BD, A \to B, B \to a, C \to 1, D \to 2\}, A)$?

Приклад 4. Яку мову задає граматика G_1 із прикладу 3 ?

Приклад 5. Яка граматика породжує множину ідентифікаторів?

Приклад 6. Яка граматика задає множину "вкладених дужок" $\{(^n)^n \mid n \geq 0\}$?

Приклад 7. Яка граматика визначає мову $\{a^n b^n c^n \mid n \geq 1\}$?

Алгоритм роботи тренажеру поділений на два блоки:

- тестування;
- виконання прикладів.

На стартовій панелі виводиться тема, назва дистанційного курсу, ПІБ автора, поточний рік, кнопка для переходу до тестування, кнопка для переходу до прикладів.

Після завершення проходження тренажеру буде виведено результат, де буде вказано кількість пройдених кроків і помилок. Знову можна вибрати наступну дію.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ємець О. О. Методичні рекомендації щодо оформлення пояснювальних записок до курсових проектів (робіт) для студентів за освітньою програмою «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки та інформаційні технології», «Комп'ютерні науки» галузь знань - 12 «Інформаційні технології» / О. О. Ємець – Полтава : РВВ ПУЕТ, 2017. – 69с.
2. Палюх Б.В. Электронное обучение в инженерном образовании / Б.В. Палюх, А. В. Твардовский, В.К. Иванов, – 2012.– Качество образования, 10, с.34–37.
3. Лисовенко Н.Н. Информационно-программная піддержка адаптивного онлайнового обучения. Монография. под ред. Л.Н. Савчук / Н.Н.Лисовенко, И.С.Белова, В.В.Викторов. – Днепропетровск: «Герда», 2014, - 78с.
4. Кузьмина М.В. Облачные технологии для дистанционного и медиаобразования: Учебно-методическое пособие / М.В. Кузьмина, Т.С. Пивоварова, Н.И. Чупраков. - Киров: Изд-во. КОГОАУ ДПО (ПК) "Институт развития образования Кировской области", 2013. - 80 с.
5. Швачич Г.Г. Сучасні інформаційно-комунікаційні технології: Навчальний посібник. / Г.Г.Швачич, В.В.Толстой, Л.М.Петречук,Ю.С.Іващенко, О.А.Гуляєва, Соболенко О.В. – Дніпро: НМетАУ, 2017. –230с
6. Автоматизація процесу програмування [Електронний ресурс] / Матеріал з Вікіпедії — вільної енциклопедії. – Режим доступу: https://uk.wikipedia.org/wiki/Автоматизація процесу програмування.
7. Інтегроване середовище розробки [Електронний ресурс] / Матеріал з Вікіпедії — вільної енциклопедії. – Режим доступу: https://uk.wikipedia.org/wiki/Інтегроване_середовище_розробки.
8. Сагиндыкова, А. С. Актуальность дистанционного образования / А. С. Сагиндыкова, М. А. Тугамбекова. — Текст : непосредственный //

Молодой ученый. — 2015. — № 20 (100). — С. 495-498. — URL: https://moluch.ru/archive/100/20703/

9. Черненко О.О. Електронний навчально-методичний посібник для самостійного вивчення навчальної дисципліни «Теорія програмування» для студентів напряму 6.040302 «Інформатика» – Режим доступу: http://tprogr.ho.ua.
10. Бабій М.С. Теорія програмування: Навчальний посібник [Електронний ресурс] / М.С. Бабій, О.П. Чекалов.– Суми: Вид-во СумДУ, 2009. – 181 с.
11. 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.
12. Базурін В. М. Середовища програмування як засіб навчання учнів основ програмування / В. М. Базурін // Інформаційні технології і засоби навчання. - 2017. - Т. 59, вип. 3. - С. 13-27. - Режим доступу: http://nbuv.gov.ua/UJRN/ITZN_2017_59_3_4.
13. Шилдт Герберт Java 8. Полное руководство; 9-е изд.: Пер. с англ. / Герберт Шилдт - М. : ООО "И.Д. Вильямс", 2015. - 1376 с.
14. 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.