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 БАКАЛАВРСЬКОЇ РОБОТИ

на тему

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ТРЕНАЖЕРА З ТЕМИ «СПОСОБИ ЗАДАННЯ МОВ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ТЕОРІЯ ПРОГРАМУВАННЯ»

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Величко Артур Олександрович
_____ «___»_____2020р.
(підпис)

Науковий керівник к. фіз.-мат. н., доц. Черненко Оксана Олексіївна
_____ «___»_____2020р.
(підпис)

ПОЛТАВА 2020р.

ЗМІСТ

ВСТУП

Стрімкий розвиток інформаційних технологій, окрім помітного зниження тимчасових і просторових бар'єрів в розповсюдженні інформації, відкрив нові перспективи у сфері освіти. Можна з упевненістю стверджувати, що в сучасному світі має місце тенденція злиття освітніх і інформаційних технологій і формування на цій основі принципово нових інтегрованих технологій навчання, заснованих, зокрема, на Інтернет-технологіях.

Інформаційні технології - процес, що використовує сукупність засобів і методів збору, обробки і передачі даних, для отримання інформації нової якості про стан об'єкту, процесу або явища.

За останні тридцять років ми стали свідками дуже бурхливого зростання математичних досліджень, пов’язаних з комп’ютерами та обчислювальною технікою. Величезний розвиток обчислювальної техніки відкрив нові горизонти для розвитку математики, який за швидкістю не має аналогів у довгій історії математики. З одного боку, нові дослідження в математиці безпосередньо ініціювали розвиток комп’ютерів і комп’ютерних наук. З іншого боку, досягнення в галузі комп’ютерних наук сприяли енергійному розвитку деяких розділів математики. Це, зокрема, стосується дискретної математики і таких її розділів як теорія формальних мов і скінченних автоматів та теорія алгоритмів. Ці області складають захоплюючу частину сучасної математики, і їм притаманний дуже динамічний характер розвитку, який повний складних завдань, що вимагають цікавих і винахідливих математичних методів. Даний посібник є гарною основою для розуміння цих областей математики. Загальновизнано, що найкращий спосіб вивчення математики ґрунтується на розв’язуванні прикладів та практичних задач. Практичний підхід подачі матеріалу не тільки допомагає вивчити методи та інструменти для розв’язання задач, а й зміцнює розуміння основних понять.

Вивчення алгоритмів і структур даних — основа будь-якого комп'ютерного курсу, не тільки для програмістів і тих, хто вивчає обчислювальну техніку. Кожен, хто користується комп'ютером, бажає, щоб він працював швидше і вирішував масштабніші задачі. Це стосується будь-якої області наукових досліджень, інженерних розрахунків і сучасних програмних систем: від моделювання системи N тіл в фізиці до розшифровки генетичного коду в молекулярній біології; від системи архітектурного проектування до моделювання літаків; від систем керування базами даних до механізмів пошуку в Інтернеті.

При написанні комп'ютерних программ досить часто реалізується метод, який уже був розроблений раніше для вирішення якої-небудь задачі. Такий метод часто не залежить від конкретної мови програмування. Саме метод, а не комп'ютерна програма, описує кроки, які необхідно виконати для вирішення задачі. Термін **алгоритм** застосовується в обчислювальній техніці для описання кінцевого, детермінованого і ефективного методу вирішення задачі, який можна реалізувати у вигляді комп'ютерно їпрограми. Алгоритм можна визначити, описавши процедуру для вирішення задачі на звичайній мові, у вигляді схеми або комп'ютерної програми.

Більшість корисних алгоритмів вимагають організації даних, які використовуються в обчисленнях.

Така організація називається **структурами даних**, і ці структури також є основними об'єктами вивчення в обчислювальній техніці. Алгоритми і структури даних працюють в тісній взаємодії. Існує думка, що структури даних існують як побічні або кінцеві продукти алгоритмів, і їх вивченняне обхідне для розуміння алгоритмів. Прості алгоритми можуть вимагати складних структур даних, і навпаки, складні алгоритми можуть використовувати прості структури даних.

Коли комп'ютер використовується для вирішення задачі, то звичайно існує декілька можливих підходів. Для невеликих задач майже не маєрізниці, який з підходів використовувати, якщо вони всі правильно

вирішують задачу. Однак у випадку великих задач швидко усвідомлюється необхідність методів, які ефективно використовують процесорний час і пам’ять. Основною причиною вивчення алгоритмів є те, що ця дисципліна в принципі дозволяє значно економити ресурси — аж до того, що стає можливим виконувати задачі, які інакше були б недоступні. Ретельна розробка алгоритму — неймовірно ефективна частина процессу вирішення великої задачі в будь-якій прикладній області.

При розробці великої або складної комп’ютерної програми необхідно витратити багато зусиль на розуміння і чітке визначення вирішуваної задачі, керування її складністю і розкладання на менші підзавдання, які неважко реалізувати.

Досить часто після такого розкладання необхідні алгоритми реалізуються елементарно. Однак в більшості випадків існує декілька алгоритмів, і вибір одного з них дуже важливий, так як значна частина системних ресурсів буде витрачена на виконання цих алгоритмів.

Використання чужих розробок в комп’ютерних системах постійно розширюється, тому слід очікувати не тільки того, що прийдеться використовувати уже розроблені алгоритми, а й того, що реалізовувати доведеться тільки невелику їх частину. Наприклад, бібліотеки Java містять реалізації величезної кількості фундаментальних алгоритмів. Однак реалізація простих варіантів базових алгоритмів допомагає краще зрозуміти їх і тому ефективніше використовувати, а також осмислено налаштовувати складні версії із бібліотек. І, що важливіше, іноді виника є необхідність в самостійній реалізації базових алгоритмів. Звичайно така необхідність виникає, якщо доводиться починати роботу в нових обчислювальних середовищах (як апаратних, так і програмних) з новими можливостями, які не враховуються в старихреалізаціях.

Темою даної дипломної роботи є «способи задання мов». Ця тема ставить перед нами завдання в знаходженні термінів і їх роз тлумачення для того, щоб зрозуміти до чого взагалі потрібен той чи інший «інструмент».

Після цього ставимо собі мету та що і в якій послідомності буде виконуватись робота.

Метою дипломної роботи є викладення інформації про те, які існують способи задання мов і їх алгоритм, мови програмування, які мови взагалі існують. Відокремити їх історію створення. Синтаксис і основні відмінності. Тема «Способи задання мов»

Структура дипломної роботи являє собою з трьох розділів:

1) Постановка задачі
2) теоретична частина
3) практична частина

Далі складений висновок, в якому описано, що дає розробка тренажеру для теми даної дипломної роботи та список використаних джерел. Обсяг дипломної роботи становить 43 сторінку включаючи титульну.

На мові програмування **C#** написано тренажер з теми «Способи задання мов».

У теоретичній частині було розглянуто основні поняття з теми «способи задання мов» і їх застосування для розробки так званого тренажеру. У практичній частині описано алгоритм роботи тренажера.

Дана тема є актуальна тим, що показує сучасний підхід до розробки програмного забезпечення. А в свою чергу сучасне програмне забезпечення полегшує життя людям в їх повсякденних потребах.

1. ПОСТАНОВКА ЗАДАЧІ

1.1. Постановка задачі для реалізації з теми «Способи задання мов» у програмуванні

Основним завданням роботи є опис способів задання мов, взятий дистанційного курсу «Теорія програмування». Планується, який розробляється в рамках роботи використовуватиметься, як складова дистанційного курсу, а отже необхідно врахувати можливість інтеграцій даної роботи до системи дистанційного навчання, на якій власне розміщуються дистанційні курси.

Розглянемо основні завдання роботи:

- Описати що таке мови програмування
- Класифікація мов програмування
- Показати приклади способів задання мов

Описати поняття:

- Категорії мов програмування
- Мови програмування та мовні процеси
- Алгоритмічна мова
- Формальна граматика мов програмування

2. Дистанційне навчання як інноваційний спосіб навчання

2.1. Поняття «Дистанційне навчання»

Дистанційне навчання — сукупність сучасних технологій, що забезпечують доставку інформації в інтерактивному режимі за допомогою використання ІКТ (інформаційно-комунікаційних технологій) від тих, хто навчає (викладачів, визначних постатей у певних галузях науки, політиків), до тих, хто навчається (студентів чи слухачів). Застосовується під час підготовки як у середніх загальноосвітніх школах і ЗВО, так і в бізнес-школах. Основними принципами дистанційного навчання є інтерактивна взаємодія у процесі роботи, надання студентам можливості самостійного освоєння досліджуваного матеріалу, а також консультаційний супровід у процесі дослідницької діяльності. Дає змогу навчатися на відстані, за допомогою диспутів експертів із кількох країн, за відсутності викладача. Основну роль у здійсненні дистанційного навчання відіграють сучасні інформаційні технології.

2.2. Історія «Дистанційного навчання»

У Європі в кінці XVIII століття, з появою регулярного і доступного поштового зв'язку, виникло «кореспондентське навчання». Історично дистанційне навчання скоріш за все у 1840 році, коли Ісаак Пітман запропонував навчання через поштовий зв'язок для студентів Англії. У 1856 році Чарльз Тюссе та Густав Лангеншейдт розпочали викладання мови заочною формою у Німеччині. Учні поштою отримували навчальні матеріали, листувалися з педагогами та здавали іспити довіреній особі або підтверджували рівень здобутої освіти у вигляді наукової роботи.

Кінець XIX століття характеризується бурхливим технологічним зростанням, наявністю телеграфу і телефону. Але достовірних фактів про їх використання в навчанні немає. У той же час триває епоха

«кореспондентського навчання», велика кількість освітніх закладів у всьому світі вели і ведуть його до сих пір.

Поява радіо і телебачення внесло зміни в дистанційні методи навчання. Це був значний прорив, аудиторія навчання зросла в сотні разів. Стали популярними навчальні телепередачі, які йшли, починаючи з 1950-х років. Однак у телебачення та радіо був істотний недолік — не було можливості забезпечити учня зворотнім зв'язком у реальному часі.

У 1969 році у Великій Британії було відкрито університет дистанційної освіти — Відкритий університет. Він був так названий, щоб показати його доступність за відносно невелику ціну та відсутність потреби часто відвідувати аудиторні заняття.

Інші відомі університети того періоду з програмами дистанційного навчання: Університет Південної Африки (статус закладу дистанційної освіти з 1946-го), Гаґенський заочний університет (Німеччина, засновано у 1974-му), INTEC-коледж (Кейптаун, ПАР з 1972-го як філіал ICS – International Correspondence Schools), Національний університет дистанційної освіти (Іспанія, рік заснування 1972), Національний технологічний університет (США, рік заснування 1984, реалізовані програми дистанційної освіти за інженерними спеціальностями).

Наприкінці 1980-х поява персональних комп'ютерів дала новий поштовх, пов'язаний зі спрощенням та автоматизацією процесу навчання. Комп'ютерні навчальні програми з'явились на перших комп'ютерах у формі ігор.

У XXI столітті доступність комп'ютерів та Інтернету роблять поширення дистанційного навчання ще простішим і швидшим. Інтернет став величезним проривом, значно більшим, ніж радіо і телебачення. З'явилася можливість спілкуватися і отримувати зворотний зв'язок від будь-якого учня, де б він не знаходився. Поширення «швидкого інтернету» дало можливість використовувати «онлайн» семінари (вебінари) для навчання.

В Україні датою офіційного початку запровадження дистанційного навчання можна вважати 21 січня 2004 року, коли наказом № 40 Міністерства освіти і науки України було затверджено «Положення про дистанційне навчання», яке поклало початок запровадженню нових технологій у галузі освіти.

Одним з перших великих центрів дистанційного навчання, що поширював свої матеріали через мережу інтернет стала Хан-Академія. ЇЇ автор Салман Хан почав викладати у вільний доступ в мережі власні відео-уроки, що створював для своєї кузени. Згодом, коли уроки отримали неабияку популярність і мільйони переглядів в мережі YouTube Хан створив окремий навчальний центр.

В Україні є декілька прикладів великих навчальних центрів з дистанційною формою навчання, що спеціалізуються на підготовці учнів до ЗНО та університетських курсах. Одним з найбільших є Prometheus — проект масових відкритих онлайн-курсів.

У 2007 році була відкрита перша в Україні державна Міжнародна українська школа, яка дає можливість учням, які проживають за межами України, здобути початкову, базову та повну середню освіту і офіційні документи державного зразка, які підтверджують освітні рівні, здобуті як дистанційно, так і екстерном. Окрім цього, в Україні функціонують декілька приватних шкіл із дистанційною формою навчання та екстернатом, у яких можуть навчатися і ті діти, які проживають в Україні, але через певні обставини не мають змоги відвідувати звичайні школи.

Актуальність проблеми розвитку дистанційної освіти викликана тим, що в сучасних умовах кардинально змінюється вимоги до спеціалістів. При цьому на центральне місце виступає не використання раніше отриманих знань, а генерація і впровадження нових ідей, що і диктує нові вимоги до підготовки кадрів. В зв’язку з цим освіта впродовж життя стає все більш необхідною, набуває нових форм та значення. Нині досягнення в області інформаційних технологій та телекомунікацій дозволяють розвиток

дистанційного навчання, як різновид безперервної освіти, яке супроводжує інформаційне суспільство, бо формує всебічно розвинену особистість, здатну орієнтуватися в інформаційному середовищі.

Важливе значення даної проблеми зумовлено тим, що з розвитком сучасних інформаційних технологій відкриваються нові перспективи для підвищення ефективності освітнього процесу. Змінюється сама парадигма освіти. Велика роль надається методам активного пізнання, самоосвіті, дистанційним освітнім програмам.

Впровадження сучасних інформаційних технологій у повсякденне життя українських громадян стало каталізатором для розвитку процесів пов'язаних з дистанційним навчанням. Інтернет, як джерело інформації, давно вже став реальністю, а розвиток телекомунікацій, без яких ця форма навчання немислима, йде швидкими темпами. Таким чином, одна з умов реалізації дистанційного навчання, а саме підготовка технічної бази, фактично поступово виконується.

Незважаючи на те, що саме технічний потенціал сучасних інформаційних технологій допомагає реалізувати одне з головних переваг дистанційного навчання – навчання на відставні, створення у навчальному закладі відповідної матеріально-технічної бази не дозволить досягти значного ефекту. Головним все-таки є наявність і розробка учбового-методичного забезпечення для самостійної роботи студентів. Електронні носії інформації дозволяють впроваджувати так звані «електроні лекції». Лекційний матеріал може бути викладений у вигляді тексту, озвучений і доповнений відеоматеріалами; мова йде про відеолекції, слайди-лекції, якісні комп'ютерні тренінги, різного роду тести та навчальні програми, додатковий ілюстративний матеріал, а також доступ у потрібний час до довідкових даних, словникових термінів. Головна перевага даних матеріалів полягає у тому, що студент зможе самостійно користуватися ними в зручний для себе час. Для дистанційного навчання особливого значення набуває наявність і саме головне якість електронних підручників, що повинні бути

по всіх дисциплінах навчального плану. Робота з електронним підручником дозволяє зробити навчальний процес індивідуальний. Кожен студент сам вибирає послідовність вивчення навчального матеріалу виходячи зі свого інтересу і можливостей.

Під час дистанційного навчання значно збільшується частка самостійної роботи студентів, а це у свою чергу призводить до зміни змісту, форм і методів навчання. Суть роботи викладача в даних умовах полягає не в читанні лекцій, а в створенні учбового-методичного забезпечення дисципліни в електронному вигляді, у постійній роботі над внесенням необхідних змін у навчальний матеріал, підборі кольорових ілюстрацій, графіків, створенні Flash-анімацій, тестів для самоконтролю.

На підставі цього дистанційне навчання мусить характеризуватися високою професійністю викладача, прагненням до співпраці з колегами та студентами на шляху досягнення високих якісних результатів освіти.

Але на шляху дистанційної освіти є чимало труднощів. Залишаються невирішеними питання проведення контролю знань студентів. При наявності якісних комунікаційних каналів іспити можна проводити за допомогою двосторонніх відеоконференцій. Але тут виникає проблема – якості зв’язку. На сайтах деяких вузів працюють системи конференцій, але це скоріше виключення, ніж правило. Інші переваги Інтернету – доступ до глобальних бібліотек, баз даних найбільших університетів, жива робота з викладачами й іншими слухачами, здача іспитів у режимі «онлайн», цифрове відео – поки що так і залишаються проблемами. Тому єдиним реально працюючим інструментом для дистанційного навчання стала електрона пошта.

Дистанційне навчання – засноване на сучасних інформаційних і комунікаційних технологіях навчання й підвищення кваліфікації. Дистанційні технології навчання можна розглядати як природний етап еволюції традиційної системи освіти від дошки з крейдою до електронної

дошки й комп’ютерних навчальних систем, від книжкової бібліотеки до електронної, від звичайної аудиторії до віртуальної аудиторії.

Ефективність дистанційного навчання заснована на тому, що ті, кого навчають, самі почувають необхідність подальшого навчання, а не піддаються тиску з боку. Дистанційна освіта стала справжньою новацією XXI століття. Віртуальний курс лекцій дозволяє скоротити або розтягти час навчання за своїм розсудом. Ефективність дистанційного навчання полягає і в тому, що можливість навчатися дистанційно не обмежує можливості навчання й удосконалюватися в професійній діяльності під час роботи на підприємстві.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Що собою являє мова програмування

Програмування — процес проектування, написання, тестування, зневадження і підтримки комп'ютерних програм. Програмування поєднує в собі елементи інженерії (існує навіть відповідна спеціальна галузь інженерії — програмна інженерія, англ. softwareengineering), фундаментальних наук (перш за все математики) і мистецтва.

У вужчому значенні програмування розглядається як кодування — реалізація у вигляді програми одного чи кількох взаємопов'язаних алгоритмів (у сучасних умовах це здійснюється з застосуванням мов програмування). У ширшому сенсі процес програмування охоплює і створення, тобто розробку, алгоритмів, і аналіз потреб майбутніх користувачів програмного забезпечення.

У широкому значенні програмування використовується у значенні створення програми дій або алгоритмів та навчання людей або пристроїв діяти за алгоритмами.

Але що ж собою являє сама мова програмування?

Мова програмування (англ. Programminglanguage) — це штучна мова, створена для передачі команд машинам, зокрема комп'ютерам. Мови програмування використовуються для створення програм, котрі контролюють поведінку машин, та запису алгоритмів.

Більш строге визначення: мова програмування — це система позначень для опису алгоритмів та структур даних, певна штучна формальна система, засобами якої можна виражати алгоритми. Мову програмування визначає набір лексичних, синтаксичних і семантичних правил, що задають зовнішній вигляд програми і дії, які виконує виконавець (комп'ютер) під її управлінням.

З часу створення перших програмованих машин було створено понад дві з половиною тисячі мов програмування. Щороку їх кількість

поповнюється новими. Деякими мовами вміє користуватись тільки невелике число їх власних розробників, інші стають відомі мільйонам людей. Професійні програмісти зазвичай застосовують в своїй роботі декілька мов програмування.

3.2. Історія мов програмування

Власне перші мови програмування з'явилися задовго до появи перших комп'ютерів. Ще в 19-му столітті існували «програмовані» ткацькі верстати та піаніно-програвачі, спосіб програмування нагадує так звані предметно-орієнтовані мови програмування. На початку 20-го століття починають використовуватись перфокарти, та механічна обробка даних. В 1930 −1940 рр. виникає лямбда-числення та машина Тюринга, які застосовували математичну абстракцію для опису алгоритмів. Лямбда-числення згодом здійснило вплив на проектування мов програмування.

В 1940 роках створюються перші електричні, двійкові комп'ютери. Вважається, що першу мову програмування високого рівня — Планкалькюль (нім. Plankalkül) розробив німець Конрад Цузе в період 1943–1945 років, але в той час вона не була реалізована і не одержала уваги. Реалізацією мови зайнялися і здійснили лише в 1998—2000 роках.

Наприкінці 40-их — початку 50-их застосовувалися інтерпретовані системи кодування, коли певні команди мови програмування кодувалися числами, які вже інтерпретувалися машинним кодом. Ці системи називалися «автоматичним програмуванням» і були простішими для програмування, ніж машинні коди, але могли мати значно меншу (до 50 разів) швидкодію, через що часто надавали перевагу машинним кодам. До таких систем належали — ShortCode (мова програмування) (англ. ShortCode) для BINAC (1949) і UNIVAC I (1952), Speedcoding (англ. Speedcoding) для IBM 701, розроблена Джоном Бекусом у 1954.

Першою широковживаною компільованою мовою став розроблений групою Джона Бекуса Фортран, анонсований 1954 року й випущений 1957

року для IBM 704. Основним призначенням Фортрану були швидкі наукові обчислення, оголошувалося, що швидкодія згенерованого компілятором коду майже не відрізнятиметься від написаного вручну машинного коду. Уже у квітні 1958 близько половини програм для IBM 704 були написані на Фортрані. Випущений у 1958 році Фортран II дозволяв незалежну компіляцію підпрограм, що дозволило створювати більші програми, оскільки низька надійність IBM 704 не дозволяла скомпілювати без збоїв велику програму (понад 300—400 рядків) одразу. Розроблений у 1960–1962 роках Фортран IV був однією з найпоширеніших мов того часу і лишався стандартною версією Фортрану до появи 1978 року Фортрану 77.

1958 року в MIT розробили LISP — першу функційну мову, яка понад чверть століття домінувала у програмуванні задач штучного інтелекту.

Наприкінці 1950-их почали розроблятися різні мови програмування. 1958 року декілька значних груп комп'ютерних користувачів у США, включаючи SHARE — групу науковців-користувачів IBM і USE (UNIVAC Scientific Exchange, група науковців-користувачів UNIVAC) запропонували ACM заснувати робочу групу зі створення універсальної мови програмування. Також ще 1955 року німецьке Товариство прикладної математики і механіки (GAMM) заснувало комітет зі створення універсальної мови програмування. В кінці травня 1958 року було проведено зустріч у Цюриху між ACM і GAMM, на матеріалах якої у грудні опубліковано «ALGOL 58 Report». На його основі було створено 3 значні реалізації — MAD (1961), NELIAC (1963), JOVIAL (1963). З них лише JOVIAL отримав поширення, ставши на чверть століття офіційною мовою програмування у Військово-морських силах США. SHARE і IBM почали створення власної реалізації ALGOL, але припинили, врахувавши витрати на створення і просування Фортрану.

Впродовж 1959 року ALGOL 58 широко обговорювався, була запропонована нотація для опису синтаксису мов програмування — форма Бекуса-Наура. 1960 року проведено чергову зустріч і опубліковано ALGOL

60 Report. ALGOL вплинув на багато мов програмування і став стандартною мовою для публікації алгоритмів, але через ряд причин не одержав широкого поширення — він був заскладним, і не було реалізацій, які підтримували його повністю, відсутність стандартного введення-виведення привела до появи різних несумісних реалізацій, деякі неоднозначності опису мови так і не були розв'язані. Також широкого вжитку уже набув Фортран, і IBM не підтримала ALGOL.

1959 року було проведено зустріч у Пентагоні для створення мови CBL (CommonBusinessLanguage), засновано комітет з його створення, і 1960 року опубліковано початкову специфікацію COBOL 60, який невдовзі став першою мовою прийнятою у Міністерстві оборони США. 1968 року COBOL було стандартизовано ANSI.

1964 року було створено спрощену мову BASIC (BeginnersAll-purposeSymbolicInstructionCode) для навчання програмуванню студентів, які переважно спеціалізувалися у вільних мистецтвах, а не технічних науках.

Тоді як науковці переважно використовували Фортран, а бізнес — COBOL, 1963 року в IBM вирішили створити універсальні платформу IBM/360 і мову програмування. У стислі терміни до 1965 року було розроблено мову PL/I, яка поєднувала можливості Фортран, ALGOL і COBOL, і виявилась заскладною, хоча і була у широкому вжитку у 1970-их у наукових і бізнес задачах, також її підмножини (PL/C, PL/CS) використовувалися для навчання програмуванню.

На початку 1960-их було створено перші мови із динамічною типізацією — APL і SNOBOL.

SIMULA 67 була першою об'єктно-орієнтованою мовою програмування.

1965 року Ніклаус Вірт і Тоні Гоар запропонували комітету з розвитку мови ALGOL свою версію, яку згодом назвали ALGOL-W і застосовували для навчання в деяких університетах. Пропозиція була відхилена через незначну кількість змін на користь значно складнішого ALGOL 68. У ALGOL 68 з'явилися визначення структур даних і динамічні масиви.

ALGOL 68 став першою мовою із формальною специфікацією, яка однак була складною для розуміння.

1971 року Вірт опублікував опис мови Pascal, яка у 70-их стала загальновживаною для навчання студентів.

1972 року Деніс Річі розробив у BellLabs мову C. Тоді ж у Марселі створено інтерпретатор мови Пролог — першої і найвідомішої мови логічного програмування. Алан Кей у Xerox PARC розробив першу широко вживану об'єктно-орієнтовану мову — Smalltalk.

3.3. Класифікація мов програмування та способи реалізації мов

Мови класифікують за такими критеріями:

Рівень абстракції

Мови програмування високого рівня оперують сутностями ближчими людині, такими як об'єкти, змінні, функції. Мови програмування нижчого рівня оперують сутностями ближчими машині: байти, адреси, інструкції. Текст програми на мові високого рівня зазвичай набагато коротший ніж текст такої самої програми на мові низького рівня, проте програма має більший розмір.

Область застосування

Універсальні та спеціалізовані. Спеціалізовані мови теж бувають Тьюрінг-повні, та все ж їх область застосування обмежена, як наприклад у мови shell.

Підтримувані парадигми програмування

Об'єктно-орієнтовані, логічні, функційні, структурні.

Імперативні мови базуються на ідеї змінної, значення якої змінюється присвоєнням. Вони називаються імперативними (лат. imperative — наказовий), оскільки складаються із послідовностей команд, які звичайно містять присвоєння змінних <назва_змінної> = <вираз>, де вираз може посилатися на значення змінних присвоєних попередніми командами.

Способи реалізації мов

Мови програмування можуть бути реалізовані як компільовані та інтерпретовані.

Програма на компільованій мові за допомогою компілятора (особливої програми) перетвориться (компілюється) в машинний код (набір інструкцій) для даного типу процесора і далі збирається в виконавчий модуль, який може бути запущений на виконання як окрема програма. Іншими словами, компілятор переводить вихідний текст програми з мови програмування високого рівня в двійкові коди інструкцій процесора.

Якщо програма написана на скриптовій мові, то інтерпретатор безпосередньо виконує (інтерпретує) вихідний текст без попереднього перекладу. При цьому програма залишається мовою оригіналу і не може бути запущена без інтерпретатора. Процесор комп'ютера, в зв'язку з цим, можна назвати інтерпретатором для машинного коду.

Поділ на компільовані і інтерпретовані мови є умовним. Так, для будь-якої традиційно компілючої мови, як, наприклад, Паскаль, можна написати інтерпретатор. Крім того, більшість сучасних «чистих» інтерпретаторів не виконують конструкції мови безпосередньо, а компілюють їх в деяке високорівневе проміжне представлення (наприклад, з розіменуванням змінних і розкриттям макросів).

Для будь-якої інтерпритуючої мови можна створити компілятор — наприклад, мова Лісп, початково інтерпретована, може компілюватися без обмежень. Створюваний під час виконання програми код може так само динамічно компілюватися під час виконання.

Як правило, скомпільовані програми виконуються швидше і не вимагають для виконання додаткових програм, так як вже переведені на машинну мову. Разом з тим, при кожній зміні тексту програми потрібно її перекомпіляція, що уповільнює процес розробки. Крім того, скомпільована програма може виконуватися тільки на тому ж типі комп'ютерів і, як правило, під тією ж операційною системою, на яку був розрахований

компілятор. Щоб створити виконуваний файл для машини іншого типу, потрібна нова компіляція.

Інтерпретовані мови володіють деякими специфічними додатковими можливостями (див. вище), крім того, програми на них можна запускати відразу ж після зміни, що полегшує розробку. Програма на скриптовій мові може бути найчастіше запущена на різних типах машин та операційних систем без додаткових зусиль.

Однак інтерпретовані програми виконуються помітно повільніше, ніж компільовані, крім того, вони не можуть виконуватися без програми-інтерпретатора.

Деякі мови, наприклад, Java та C #, перебувають між компільованими і інтерпретованими. А саме, програма компілюється не в машинну мову, а в машинно-незалежний код низького рівня, байт-код. Далі байт-код виконується віртуальною машиною. Для виконання байт-коду зазвичай використовується інтерпретація, хоча окремі його частини для прискорення роботи програми можуть бути трансльовані в машинний код безпосередньо під час виконання програми за технологією компіляції «на льоту» (Just-in-timecompilation, JIT). Для Java байт-код виповнюється віртуальною машиною Java (JavaVirtualMachine, JVM), для C# — CommonLanguageRuntime.

Подібний підхід у деякому сенсі дозволяє використовувати плюси як інтерпретаторів, так і компіляторів. Слід згадати, що є мови, які мають і інтерпретатор, і компілятор (Форт (Forth)).

Мови програмування діляться в свою чергу на мови низького рівня та мови високого рівня.

3.4. Мови і граматики. Перший спосіб задання мов. Регулярні вирази.

Алфавіт – це скінченна множина символів. Позначатимемо його X .

Приклади:

- Множина $\{0,1\}$ являє собою бінарний алфавіт.

- ASCII і Unicode є прикладами комп'ютерних алфавітів.

Рядком у даному алфавіті називають скінченну послідовність символів з цього алфавіту. У теорії мов рядок ще називають ланцюжком, реченням, словом.

Символ ε позначає порожній рядок. Порожній рядок має нульову довжину.

Терміни “підрядок” і “підпослідовність” будемо розрізняти. Підрядок одержують видаленням початку і (або) кінця з рядка, підпослідовність – видаленням декількох символів, не обов'язково послідовних.

Множина всіх скінченних слів у алфавіті X позначається X^*. Зауважимо, що вона нескінченна. Вона містить порожнє слово – послідовність довжиною 0, позначену буквою ε . Множину $X^* \setminus \{\varepsilon\}$ позначимо X^+, а слово вигляду $ww..w$, де слово w із X^+ записано n разів – w^n. Вважатимемо, що $w^0 = \varepsilon$.

Довільна підмножина множини X^* називається формальною мовою. Далі вона буде називатися просто мовою. Мова – це довільна множина рядків у деякому алфавіті.

Приклади:

1. Множина всіх слів у алфавіті $\{a\}$ позначається $\{a\}^* = \{\varepsilon, a, aa, aaa, ...\} = \{a^n \mid n \geq 0\}$. $\{a^n \mid n -$ непарне$\}$ позначає множину, або мову слів непарної довжини в алфавіті $\{a\}$; обидві мови нескінченні.

2. Ідентифікатор є послідовністю букв і цифр, що починається буквою. Множина всіх ідентифікаторів у алфавіті $X = \{a, b, 1\}$ нескінченна. Якщо записати їх за зростанням довжини, то початок буде таким: $\{a, b, a1, aa, ab, b1, ba, bb, ...\}$.

Задача перевірки, чи належить слово w мові L, називається задачею належності, або проблемою слів. Як правило, множина L задається певним скінченним описом, що визначає не тільки її саму, а й структуру її елементів.

Задача належності розв'язується найчастіше шляхом перевірки, чи має слово відповідну структуру, тобто шляхом синтаксичного аналізу, або розпізнавання. Наприклад, структура всіх можливих синтаксично правильних Паскаль-програм визначається скінченною та відносно невеликою сукупністю БНФ. Саме на її основі будуються синтаксичні аналізатори в трансляторах, тобто програми аналізу синтаксичної правильності вхідних програм.

Формальні мови розглядатимуться далі як мови, задані саме скінченним описом. Отже, головним у вивченні формальних мов стає засіб їх задання.

Над мовами можливі операції, що породжують нові мови. Означимо ***регулярні операції*** над мовами: об'єднання, катенацію та ітерацію. Нехай L_1, L_2, L позначають довільні мови в алфавіті X. Перелік основних операцій наводиться у табл. 3.1.

Таблиця 3.1 Операції над мовами

Операція	Визначення
Об'єднання L_1 і L_2: $L_1 \cup L_2$	$L_1 \cup L_2 =$ $\{w \mid w \in L_1$ або $w \in L_2\}$.
Катенація L_1 і L_2: L_1L_2	$L_1L_2 = \{vw \mid v \in L_1, w \in L_2\}$.
Замикання Кліні, або ітерація $L : L^*$.	$L^* = \bigcup_{i=0}^{\infty} L^i = \{w^i \mid w \in L$ за $i \geq 0\}$ L^* означає 0 або більше конкатенацій L.
Позитивне замикання $L : L^+$.	$L^+ = \bigcup_{i=1}^{\infty} L^i = \{w^i \mid w \in L$ за $i > 0\}$ L^+ означає одну або більше конкатенацій L.

Вираз $L_1 \cup L_2$ позначає ***об'єднання*** L_1 і L_2 – мову $\{w \mid w \in L_1$ або $w \in L_2\}$. Наприклад, $\{a, ab\} \cup \{a, b, ba\} = \{a, b, ab, ba\}$.

Катенацією слів v і w називається дописування w після $v : vw$. Вираз L_1L_2 позначає ***катенацію мов*** – мову $\{vw \mid v \in L_1, w \in L_2\}$. Так, за

$L_1 = \{a, bc\}, L_2 = \{x, y\}$ катенація $L_1 L_2 = \{ax, bcx, ay, bcy\}$, за

$L_1 = \{a, ab\}, L_2 = \{\varepsilon, b\}$ катенація $L_1 L_2 = \{a, ab, abb\}$.

Від катенації походить ***піднесення до степеня***: $L^0 = \{\varepsilon\}, L^i = L^{i-1} L$ за $i > 0$. Так, вираз $\{\varepsilon, a, aa\}^2$ задає мову $\{\varepsilon, a, aa, aaa, aaaa\}$.

Вираз L^* позначає ***ітерацію*** мови L – мову $\{w^i \mid w \in L$ за $i \geq 0\}$, тобто $\{\varepsilon\} \cup L \cup L^{2 \cup \ldots}$. Зазначимо, що ітерація не подається жодним скінченним виразом з операціями катенації та $\cup$ і тому *не є похідною від них*. Якщо в мові L є непорожнє слово, то мова L^* нескінченна. Наприклад, вираз $\{ab\}^*$ задає мову $\{\varepsilon, ab, abab, ababab, \ldots\}, \{a, b\}\{a, b, 1\}^*$ – множину ідентифікаторів у алфавіті $\{a, b, 1\}$.

Регулярні вирази й задані ними регулярні мови означимо індуктивно. Вирази $\varnothing$, ε та a при $a \in X$ є ***регулярними*** в алфавіті X і задають відповідно ***регулярні мови*** $\varnothing, \{\varepsilon\}, \{a\}$. Якщо r_1 і r_2 – регулярні вирази, що задають регулярні мови L_1 і L_2, то вирази $(r_1), r_1 + r_2, r_1 r_2, r_1^*$ є регулярними й задають відповідно регулярні мови $L_1, L_1 \cup L_2, L_1 L_2, L_1^*$.

Так безпосередньо переходимо до другого із способів задання мов – ***формальних граматик.***

3.5. Другий спосіб задання мов. Формальні граматики.

Розглянемо дане 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.

Формально **граматика** 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, $V_T \cap V_N = \varnothing$;

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 $V = V_T \cup V_N$;

S – стартовий символ (або аксіома).

Приклади:

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

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

Тут $P = \{S \to xSy, S \to xy\}$.

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:
$S \Rightarrow xS \Rightarrow xxS \Rightarrow xxyB \Rightarrow xxyyB \Rightarrow xxyyy$.

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 $v_1 \to w_1(u_1 \mid u_2)w_2$.

Наглядний приклад: 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.

Як Хомський задавав граматики. Він виділяв 4 класи граматик. Граматики 0-го типу, загального вигляду, або рекурсивно-перерелічувані визначаються як граматики, що відповідають нашому визначенню без обмежень на типи продукцій. Це найбільш загальний клас, інші граматики можуть бути отримані накладенням обмежень на продукції граматик 0-го типу. Граматики 0-го типу еквівалентні машинам Тьюринга.

Перше обмеження: задати, щоб для всіх продукцій $\alpha \to \beta$ довжина рядка α, обчислена в кількості символів, була не більше довжини рядка β.

Граматики, усі продукції яких задовольняють дане обмеження, називаються граматиками 1-го типу, або контекстозалежними. Граматики 1-го типу еквівалентні лінійно-обмеженим автоматам.

Якщо, крім уже названого обмеження, у лівій частині продукції повинен знаходитися тільки один нетермінал, граматика називається граматикою 2-го типу або контекстовільною граматикою. Приклади G_1, G_2 представляють 2-й тип. У контекстовільних граматиках зручно дозволити продукцію $S \to \varepsilon$, хоча, строго кажучи, вона не дозволена навіть у контекстозалежних граматиках. Граматики 2-го типу еквівалентні магазинним автоматам (push-down).

Останній клас граматик – граматики 3-го типу, або регулярні (автоматні) граматики. Мова такої граматики називається регулярною. Регулярну мову можна визначити регулярним виразом і навпаки. Регулярні мови і регулярні вирази еквівалентні скінченним автоматам.

Ієрархія Хомського є включаючою (рис 4.1).

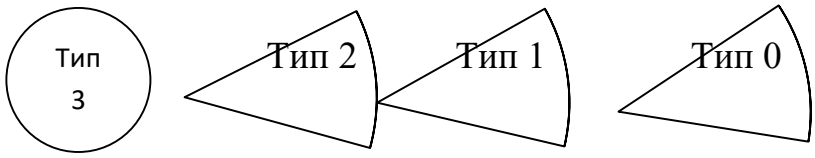

Рисунок 4.1 Ієрархія Хомського для граматик

Формальні мови класифікуються відповідно до типів граматик, якими вони задаються. Однак та сама мова може бути задана різними граматиками, що належать до різних типів. У такому випадку вважається, що мова належить до найбільш простого з них. Так, мова, описана граматикою з фразовою структурою, контекстозалежною і контекстовільною граматиками, буде контекстовільною.

Найбільш складні – мови загального вигляду (сюди можна віднести природні мови), найпростіші – регулярні мови.

4. ПРАКТИЧНА ЧАСТИНА

4.1. Первинні поняття щодо регулярних виразів

Якщо Вам колись доводилося працювати з командним рядком, Ви, ймовірно, використовували маски імен файлів. Наприклад, щоб видалити всі файли в поточній директорії, які розпочинаються з букви “d”, можна написати rm d*.

Регулярні вирази є схожим, але набагато сильнішим інструментом для пошуку рядків, перевірки їх на відповідність певному шаблону та іншої подібної роботи. Англомовна назва цього інструмента – RegularExpressions чи просто RegExp. Інакше кажучи, регулярні вирази – спеціальна мова для опису шаблонів рядків.

Реалізація цього інструмента розрізняється в мовах програмування, хоч і не істотно. У цій статті ми орієнтуватимемося передусім на реалізацію PerlCompatibleRegularExpressions.

Засади синтаксису:

Зазначимо, що будь-який рядок сам по собі є регулярним виразом. Так, виразу Хаха, очевидно, відповідатиме рядок ” Хаха” і тільки він. Регулярні вирази є регістрозалежними, тому рядок “хаха” (з маленької літери) вже не відповідатиме виразу вище.

Проте вже тут слід бути акуратним – як і будь-яка мова, регулярні вирази мають спецсимволи, які треба екранувати. Ось їх список: . ^ $ * + ? { } [ ] | ( ). Екранування здійснюється у звичайний спосіб – додаванням перед спецсимволом.

Набір символів:

Припустимо, ми хочемо знайти в тексті всі вигуки, що означають сміх. Просто Хаха нам не підійде – адже під нього не потраплять ” Хехе”, ” Хохо” і ” Хихи”. До того ж, проблему з регістром першої літери треба якось розв’язати.

Тут нам на допомогу прийдуть набори – замість вказівки конкретного символу, ми можемо записати цілий список, і якщо в досліджуваному рядку на вказаному місці стоятиме будь-який з названих символів, рядок вважатиметься відповідним. Набори записуються в квадратних дужках – патерну [abcd] відповідатиме будь-який із символів “a”, ” b”, ” c” або “d”.

Усередині набору більша частина спецсимволів не потребує екранування, проте використання \ перед ними не вважатиметься помилкою. Як і раніше, необхідно екранувати символи “\” і “^”, і, бажано “]” (так, [][] означає будь-який із символів “]” чи “[“, тоді як [[]x] – винятково послідовність “[x]”). Незвичайна, на перший погляд, поведінка регулярок із символом “]” насправді визначається відомими правилами, але набагато легше просто екранувати цей символ, ніж їх запам’ятовувати. Крім цього, екранувати треба символ “-“, він використовується для завдання діапазонів (див. нижче).

Якщо відразу після [записати символ ^, то набір набуде зворотного значення – відповідним вважатиметься будь-який символ, крім вказаних. Так, патерну [^xyz] відповідає будь-який символ, крім, власне, ” x”, ” y” або “z”.

Отже, застосовуючи цей інструмент до нашого випадку, якщо ми напишемо [Xx][аоие]x[аоие], то кожен з рядків ” Хаха”, ” хехе”, ” хихи” і навіть ” Хохо” відповідатимуть шаблону.

Зумовлені класи символів:

Для деяких наборів, що часто використовуються, є спеціальні шаблони. Так, для опису будь-якого пробільного символу (пропуск, табуляція, перенесення рядка) використовується /s, для цифр – /d, для символів латиниці, цифр і підкреслення ” _” – /w.

Якщо необхідно описати взагалі будь-який символ, для цього використовується крапка – .. Якщо вказані класи написати з великої літери (S, D, W) те вони змінюють своє значення на протилежне – будь-який

непробільний символ, будь-який символ, який не є цифрою, і будь-який символ, крім латиниці, цифр або підкреслення відповідно.

Також за допомогою регулярних виразів є можливість перевірити положення рядка відносно іншого тексту. Вираз /b означає границю слова, /B – не границю слова, ^ – початок тексту, а $ – кінець. Так, за патерном Java у рядку “JavaandJavaScript” знайдуться перші 4 символи, а за патерномbJavaB – символи з 10-го по 13-й (у складі слова ” JavaScript”).

Діапазони

У Вас може виникнути необхідність позначити набір, до якого входять літери, наприклад, від ” б” до ” ф”. Замість того, щоб писати [бвгдежзиклмнопрстуф] можна скористатися механізмом діапазонів і написати [б-ф]. Так, патерну x[0-8a - F][0-8a - F] відповідає рядок “xA6”, але не відповідає “xb9” (по-перше, через те, що в діапазоні вказані тільки великі літери, по-друге, через те, що 9 не входить у проміжок 0-8).

Механізм діапазонів особливо актуальний для російської мови, адже для неї немає конструкції, аналогічної /w. Щоб позначити всі літери російського алфавіту, можна використати патерн [а-яА-ЯеЕ]. Зверніть увагу, що літера “ё” не входить до загального діапазону літер, тому її треба вказувати окремо.

Квантифікатори (вказівка кількості повторень)

Повернемося до нашого прикладу. Якщо у “вигуку, що сміється”, буде більше однієї голосної між літерами “х”, наприклад ” Хаахаааа”? Наша стара регулярка вже не зможе нам допомогти. Тут нам доведеться скористатися квантифікаторами.

Квантифікатор	Число повторень	Приклад	Відповідні рядки
{n}	Рівно n разів	Ха{3}хаr	Хаааха
{m, n}	Від m до n включно	Ха{2,4}ха	Хаа, Хааа, Хаааааха
{m,}	Не менше m	Ха{2,}ха	Хааха, Хаааха, Хаааааха і т. д.
{,n}	Не більше n	Ха{,3}ха	Хха, Хаха, Хааха, Хаааха

Зверніть увагу на те, що квантифікатор застосовується тільки до символу, який стоїть перед ним.

Деякі часто використовувані конструкції отримали в мові регулярних виразів спеціальні позначення:

Квантифікатор	Аналог	Значення
?	{0,1}	Нуль або одне входження
*	{0,}	Нуль або більше
+	{1,}	Одне або більше

Таким чином, за допомогою квантифікаторів ми можемо поліпшити наш шаблон для вигуків до [Хх][аоеи]+х[аоеи]*, і він зможе розпізнавати рядки ” Хааха”, ” хееееех” і ” Хихии”.

Це відбувається через те, що за умовчанням квантифікатор працює за т.зв. *жадібним* алгоритмом – намагається повернути якомога довший рядок, що відповідає умові. Розв’язати проблему можна двома способами. Перший – використати вираз <[^>]*>, який заборонить вважати вмістом тега праву кутову дужку. Другий – оголосити квантифікатор не жадібним, а *ледачим*. Робиться це за допомогою додавання праворуч до

квантифікатора символу ?. Тоді для пошуку всіх тегів вираз звернеться до <.*?>.

4.2. Первинні поняття щодо граматики

Словник чи алфавіт – це кінцева множина неподільних у поточному розрізі символів. Символи, що входять до алфавіту називають літерами алфавіту.

Наприклад: алфавіт A={a,b,c,+,1} містить 5 літер, а алфавіт B={00,01,10,11} містить 4 літери, кожна з яких складається з двох символів.

Ланцюжком символів або словом в алфавіті називається будь-яка скінчена послідовність літер цього алфавіту, а кількість літер, з яких складається слово, називають довжиною слова. Порожнім ланцюжком називається ланцюжок, що не містить жодного символу. Для позначення його будемо використовувати символ e.

Більш формально ланцюжок символів в алфавіті V визначається у такий спосіб.

1. e - ланцюжок в алфавіті V.
2. Якщо a- ланцюжок в алфавіті V і a – символ цього алфавіту, то aa – ланцюжок в алфавіті V.
3. b - ланцюжок в алфавіті V тоді і тільки тоді, коли він є такий в силу п.п.1 та 2.

Якщо a і b ланцюжки, то ab - називається конкатенацією (чи зчепленням) ланцюжків a і b.

Наприклад, якщо:

a = ab і b=cd, то ab=abcd.

Для будь-якого ланцюжка a вірний вираз: ae=ea=a.

Реверсом ланцюжка a називається ланцюжок, символи якого записані в зворотному порядку.

Реверс ланцюжка a будемо позначати aR. Наприклад, якщо:

a=abcdef, то aR=fedcba.

Для порожнього ланцюжка:

e=eR.

n-мступенем ланцюжка a (будемо позначати an) називається конкатенація n ланцюжків a:

a0=e; an=aan-1=an-1a.

Довжину ланцюжка a будемо позначати $|\alpha|$. Довжина e=0.

Мова в алфавіті V – це підмножина скінчених ланцюжків в цьому алфавіті. Позначимо через V* множину, що містить усі ланцюжки в алфавіті V, включаючи порожній ланцюжок e.

Наприклад, якщо:

V={0,1}, то V*={e, 0, 1, 00, 01, 11, 10, 000, 001, 011,…}.

Позначимо через V+ множину, що містить усі ланцюжки в алфавіті V крім порожнього ланцюжка e.

Отже:

$V^* = V^+ \cup \varepsilon.$

Декартовим добутком A´B множин A та B називається множина:

$\{(a,b) \mid a \in A,\ \ b \in B\}.$

Зрозуміло, що кожна мова в алфавіті V є підмножиною множини V*.

Відомо кілька різних способів опису мов. Один з них використовує породжуючі граматики.

Формальною породжуючою граматикою G називають наступну четвірку об’єктів:

G=(VT, VN, P, S),де: VT – алфавіт термінальних символів (терміналів). Літери цього алфавіту називаються термінальними символами, з них будуються ланцюжки, що породжуються граматикою;

VN – алфавіт нетермінальних символів (не терміналів), що не перетинається з VT. Літери цього алфавіту використовуються при побудові ланцюжків. Вони можуть входити у проміжкові ланцюжки, але не повинні входити у результат породження;

P – множина правил висновку чи породжуючих правил наступного виду: $\alpha \to \beta$, де a і b - ланцюжки, побудовані з літер алфавіту $VN \cup VT$, який має назву повний алфавіт (словник) граматики G.

S – початковий символ граматики, $S \in VN$.

У множині правил граматики можуть зустрічатися правила з пустою правою частиною $\alpha \to$, тому, для уникнення невизначеності домовимось записувати таке правило у вигляді: $\alpha \to \varepsilon$.

Для запису правил висновку з однаковими лівими частинами:

$\alpha \to \beta_1, \alpha \to \beta_2, \alpha \to \beta_3, \ldots, \alpha \to \beta_n$

будемо користуватися скороченим записом:

$\alpha \to \beta_1 \mid \beta_2 \mid \beta_3 \mid \ldots \mid \beta_n$.

Кожне bi (i = 1, 2, ..., n) будемо називати альтернативою правила висновку з ланцюжка a.

Приклад граматики:

G1= ({0,1}, {A, S}, P, S),

де P складається з правил:

S → 0A1

0A → 00A1

A → e.

Визначення.

Нехай r = t → g правило граматики G і a=c/ t c// - ланцюжок символів, причому $\chi', \chi'' \in (VT \cup VN)^*$.

Тоді ланцюжок b=c/ g c// може бути отриманим з ланцюжка a шляхом застосування правила r (тобто заміною в ланцюжку t на g).

У цьому випадку кажуть, що ланцюжок b безпосередньо виведений з ланцюжка a та позначають $\alpha \Rightarrow \beta$.

Наприклад:

ланцюжок 00A11 безпосередньо виведений з 0A1 у граматиці G1.

Якщо задана послідовність ланцюжків W = (v0, v1, … vn) таких, що існує послідовність безпосередніх виведень:

v0⇒ v1, v1⇒ v2, ..., vn-1⇒ vn,

то таку послідовність називають виведенням довжини n vn з v0 в граматиці G та позначають v0⇒ *vn.

Наприклад:

S⇒ 000A111 у граматиці G1, тому що існує виведення S→0A1→00A11→ 000A111. Довжина виведення – 3.

Мовою, що породжена граматикою G=(VT, VN, P, S), називається множина L(G)= {a∈VT*½S⇒ a}.

Іншими словами, L(G), це всі ланцюжки в алфавіті VT, що виведені з S за правилами P.

Наприклад:

L(G1)={0n1n |n>0}.

Ланцюжок a∈(VT◡VN)*, для якої S⇒ a, називається сентенціальною формою в граматиці G=(VT, VN, P, S).

Таким чином, мову, яку складає граматика можна визначити як множину термінальних сентиціальних форм.

4.3. Алгоритм роботи тренажера

Перед користувачем відкриється вікно програми тренажера. У вікні знаходиться дві кнопки: «Теоретичне частина» та «Практичне тестування». Якщо користувач обере пункт «Теоретична частина», то перед ним з’явиться весь теоретичний матеріал, що допоможе засвоїти дану тему. Якщо користувач обере пункт «Практичне тестування», з’явиться вікно в якому буде викладений матеріал у формі тестів, та на вибір декілька задач. Після проходження тестування перед користувачем з’явиться вікно оцінювання, та буде кнопка, що поверне користувача до головного вікна. У головному вікні користувач знов може обрати. Ознайомитись з теоретичною частиною, або пройти практичне тестування.

Крок 0. Відображена умова: Мова $\{x^n y^n \mid n > 0\}$ описується граматикою

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

Тут $P = \{S \to xSy, S \to xy\}$.

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

Тут набір продукції P має вигляд

1	$S=>xS$	4	$S=>y$
2	$S=>yB$	5	$B=>yB$
3	$S=>x$	6	$B=>y$

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

Записати породження даної граматики покроково.

Який спосіб генерує рядок $xxyyy$?

Дана умова буде відображається впродовж усього вирішення завдання.

Крок 1. Виводиться умова: Якою буде початкова точка, для рішення вище наведеного виразу? Оберіть правильний варіант:

А) *S*;

Б) *B*;

В) *x*;

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 2 Виводиться умова: Який символ буде обрано далі, якщо початковий символ «S», виходячи з умови вище наведеної задачі? Оберіть правильний варіант:

А) Буде обрано пункт 1, що матиме вигляд: $S => xS$

Б) Буде обрано пункт 2, що матиме вигляд: $S => yB$

В) Буде обрано пункт 3, що матиме вигляд: $S => x$

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 3. Виводиться умова: Виходячи з прикладу $S => xS$ та умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити? Оберіть правильний варіант:

А) $S => xS$, потрібно замінити S на другий номер у таблиці, що матиме вигляд $S => x\underline{S} => xyB$

Б) $S => xS$, потрібно замінити S на перший номер у таблиці, що матиме вигляд $S => x\underline{S} => xxS$

В) $S => xS$, потрібно замінити S на третій номер у таблиці, що матиме вигляд $S => x\underline{S} => xx$

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 4. Виводиться умова: Виходячи з прикладу $S => x\underline{S} => xxS$ та умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити? Оберіть правильний варіант:

А) $S => x\underline{S} => xxS$, потрібно замінити S на перший номер у таблиці, що матиме вигляд $S => xS => xx\underline{S} => xxxS$

Б) $S => x\underline{S} => xxS$, потрібно замінити S на другий номер у таблиці, що матиме вигляд $S => xS => xx\underline{S} => xxyB$

В) $S => x\underline{S} => xxS$, потрібно замінити S на третій номер у таблиці, що матиме вигляд $S => xS => xx\underline{S} => xxx$.

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 5. Виводиться умова: Виходячи з прикладу *S =>xS=>xxS =>xxyB* та умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити? Оберіть правильний варіант:

А) *S =>xS=>xxS =>xxyB,* потрібно замінити *B* на шостий номер у таблиці, що матиме вигляд *S =>xS=>xxS =>xxyB=>xxyy*

Б) *S =>xS=>xxS =>xxyB,* потрібно замінити *B* на п'ятий номер у таблиці, що матиме вигляд *S =>xS=>xxS =>xxyB=>xxyyB*

В) Подальшого розв'язку даного виразу не існує.

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 6. Виводиться умова: Виходячи з прикладу *S =>xS=>xxS =>xxyB=>xxyyB* та умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити? Оберіть правильний варіант:

А) *S =>xS=>xxS =>xxyB=>xxyyB,* потрібно замінити *B* на п'ятий номер у таблиці, що матиме вигляд *S =>xS=>xxS =>xxyB=>xxyyyB*

Б) *S =>xS=>xxS =>xxyB=>xxyyB,* потрібно замінити *B* на шостий номер у таблиці, що матиме вигляд *S =>xS=>xxS =>xxyB=>xxyyy*

В) Подальшого розв'язку даного виразу не існує.

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 7. Виводиться повідомлення про завершення проходження тренажера та шала оцінювання. Пропонується пройти тренажер знову або завершити його. Якщо вибрано повторне проходження, то відбувається перехід на крок 0. Також можна обрати інший приклад для проходження.

Крок 0. Відображена умова: Мова *G= ({a,b,c},{S,A,B,C},P,S)*

Тут набір продукції *P* має вигляд

1	*S=>aSBC*
2	*S=>abC*
3	*CB=>BC*
4	*B=>aSBC*
5	*bB=>bb*
6	*BC=>bc*
7	*cC=>cc*
8	*BC=>S*
9	*bC=>c*
10	*C=>B*
11	CC=>cB

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

Знайти кінцеве значення aabbcc. Та записати породження даної граматики покроково.

Крок 1. Виводиться умова: Якою буде початкова точка, для рішення вище наведеного виразу? Оберіть правильний варіант:

А) *S*;

Б) *B*;

В) *x*;

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 2. Виводиться умова: Виходячи з початкового символу *S,* та умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити, для вірної побудови породження? Оберіть правильний варіант:

А) Маючи початкову точку *S,* для побудови породження її потрібно замінити на символи у таблиці під номером один, що матиме вигляд – *S=>aSBC.*

Б) Маючи початкову точку *B,* для побудови породження її потрібно замінити на символи у таблиці під номером два, що матиме вигляд – *B=>aSBC*

В) Маючи початкову точку *S,* для побудови породження її потрібно замінити на символи у таблиці під номером два, що матиме вигляд – *S=>aBC*

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 3. Виводиться умова: Виходячи з породження *S=>aSBC,* та умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити, для вірної побудови породження? Оберіть правильний варіант:

А) У виразі *S=>aSBC* потрібно замінити літеру *S,* на номер один у таблиці, що матиме вигляд: *S=>aSBC=>aaSBCBC*

Б) У виразі *S=>aSBC* потрібно замінити літери *BC,* на номер шість у таблиці, що матиме вигляд: *S=>aSBC=>aSbc.*

В) У виразі *S=>aSBC* потрібно замінити літеру *S,* на номер два у таблиці, що матиме вигляд: *S=>aSBC=>aabCBC.*

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 4. Виводиться умова: Виходячи з породження *S=>aSBC=>aabCBC,* та умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити, для вірної побудови породження? Оберіть правильний варіант:

А) У виразі *S=>aSBC=>aabCBC* потрібно замінити літери *BC,* на номер шість у таблиці, що матиме вигляд: *S=>aSBC=>aabCBC=>aabCbc*

Б) У виразі *S=>aSBC=>aabCBC* потрібно замінити літери *CB,* на номер три у таблиці, що матиме вигляд: *S=>aSBC=>aabCBC=>aabBCC*

В) У виразі *S=>aSBC=>aabCBC* потрібно замінити літери *BC,* на номер вісім у таблиці, що матиме вигляд: *S=>aSBC=>aabCBC=>aabSC*

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 5. Виводиться умова: Виходячи з породження *S=>aSBC=>aabCBC=>aabBCC,* та умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити, для вірної побудови породження? Оберіть правильний варіант:

А) У виразі *S=>aSBC=>aabCBC=>aabBCC* потрібно замінити літери *BC,* на номер шість у таблиці, що матиме вигляд: *S=>aSBC=>aabCBC=> aabBCC=>aabbcC*

Б) У виразі *S=>aSBC=>aabCBC=>aabBCC* потрібно замінити літери b*B,* на номер п’ять у таблиці, що матиме вигляд: *S=>aSBC=>aabCBC=> aabBCC=>aabbCC*

В) У виразі *S=>aSBC=>aabCBC=>aabBCC* потрібно замінити літеру *B,* на номер один у таблиці, що матиме вигляд: *S=>aSBC=>aabCBC=> aabBCC=>aabaSBCCC*

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 6. Виводиться умова: Виходячи з породження *S=>aSBC=>aabCBC=> aabBCC=>aabbCC,* та умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити, для вірної побудови породження? Оберіть правильний варіант:

А) У виразі *S=>aSBC=>aabCBC=> aabBCC=>aabbCC* потрібно замінити літери *bC,* на номер дев'ять у таблиці, що матиме вигляд: *S=>aSBC=>aabCBC=> aabBCC=>aabbCC=>aabbcC*

Б) У виразі *S=>aSBC=>aabCBC=> aabBCC=>aabbCC* потрібно замінити літеру *C,* на номер десять у таблиці, що матиме вигляд: *S=>aSBC=>aabCBC=> aabBCC=>aabbCC=>aabbBC*

В) У виразі *S=>aSBC=>aabCBC=> aabBCC=>aabbCC* потрібно замінити літери *CC,* на номер одинадцять у таблиці, що матиме вигляд: *S=>aSBC=>aabCBC=> aabBCC=>aabbCC=>aabbcB*

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 7. Виводиться умова: Виходячи з породження *S=>aSBC=>aabCBC=> aabBCC=>aabbCC=>aabbcC,* та умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити, для вірної побудови породження? Оберіть правильний варіант:

А) У виразі *S=>aSBC=>aabCBC=> aabBCC=>aabbCC=>aabbcC* потрібно замінити літеру *C,* на номер десять у таблиці, що матиме вигляд: *S=>aSBC=>aabCBC=> aabBCC=>aabbCC=>aabbcC =>aabbcB*

Б) У виразі *S=>aSBC=>aabCBC=> aabBCC=>aabbCC=>aabbcC* потрібно замінити літери *cC,* на номер сім у таблиці, що матиме вигляд: *S=>aSBC=>aabCBC=> aabBCC=>aabbCC=>aabbcC =>aabbcc*

В) Рішення не існує, цикл може продовжуватись до безкінечності.

Якщо відповідь вірна то відбувається перехід на наступний крок. Якщо відповідь не вірна відображається повідомлення про помилку: «Відповідь не вірна, оберіть інший варіант!!!»

Крок 8. Виводиться повідомлення про завершення проходження тренажера та шала оцінювання. Пропонується пройти тренажер знову або завершити його. Якщо вибрано повторне проходження, то відбувається перехід на крок 0. Також можна обрати інший приклад для проходження.

4.4. Блок-схема роботи алгоритму

На рисунку 4.1 зображена блок-схема алгоритму роботи програми-тренажера.

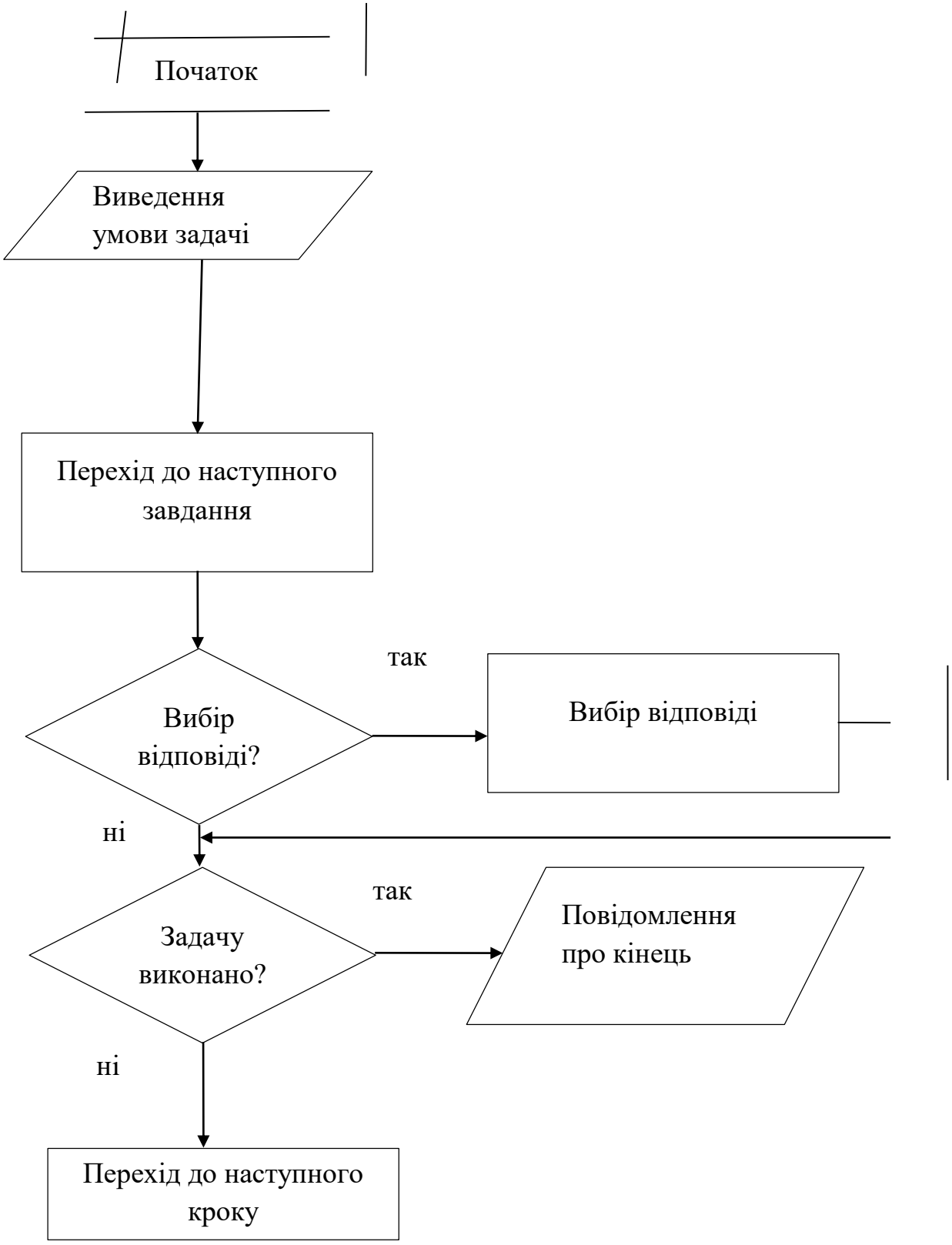

Рис. 4.1 Блок-схема алгоритму роботи тренажеру

Дана блок-схема є оптимальним варіантом для роботи тренажеру і не потребує дороблення.

4.5. Опис процесу програмної реалізації тренажеру

Програмний продукт «Тренажер з теми «Способи задання мов» дистанційного навчального курсу «Теорії програмування» розроблений на платформі .NET Framework 2.0 з використанням мови програмування C#.

Для універсальності, уніфікації відображення та простоти роботи більшість інформації відображатиметься в форматі HTML за допомогою стандартного компонента WebBrowser.

У Додатку А знаходяться створені форми.

Класи «test1» та «test2» реалізують роботу з конкретними тестуваннями. Вони зберігають інформацію про тести, варіанти та правильні відповіді. Код класу «test1»

```
class test1 : testOne
{
   public test1()
   {
      string umova = "<b>Мова {x<sup>n</sup>y<sup>n</sup>|n>0} описується граматикою<br>" +
         "G<sub>1</sub>=({x,y},{S},P,S).<br>" +
         "Тут P={S→xSy,S→xy}.<br>" +
         "Граматикою для мови {x<sup>m</sup>y<sup>n</sup>|m,n≥0} G<sub>2</sub>=({x,y},{S,B},P,S).<br>" +
         "Тут набір продукції P має вигляд" +
         "<table class='b' style='font-weight: bold;'>" +
            "<tr><td>1<td>S=>xS<td>4<td>S=>y" +
            "<tr><td>2<td>S=>yB<td>5<td>B=>yB" +
            "<tr><td>3<td>S=>x<td>6<td>B=>y" +
         "</table>" +
         "Оскільки порожній рядок також належить мові, у набір P також входить продукція S→ε.<br>" +
         "Записати породження даної граматики покроково.<br>" +
         "Який спосіб генерує рядок xxyy ?<br><br></b>";
      //1
      this.addTest(new testData(umova + "Якою буде початкова точка, для рішення вище наведеного виразу? Оберіть правильний варіант:" +
```

```
                "<p style='text-align:left;'>" +
                "<br>А) S;" +
                "<br>Б) B;" +
                "<br>В) x;</p>",
                1));
            //2
            this.addTest(new testData(umova + "Який символ буде обрано далі, якщо початковий символ «S», виходячи з умови вище наведеної задачі? Оберіть правильний варіант:" +
                "<p style='text-align:left;'>" +
                "<br>А) Буде обрано пункт 1, що матиме вигляд: S => xS" +
                "<br>Б) Буде обрано пункт 2, що матиме вигляд: S => yB" +
                "<br>В) Буде обрано пункт 3, що матиме вигляд: S => x</p>",
                1));
            //3
            this.addTest(new testData(umova + "Виходячи з прикладу S => xS та  умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити? Оберіть правильний варіант:" +
                "<p style='text-align:left;'>" +
                "<br>А) S => xS, потрібно замінити S на другий номер у таблиці, що матиме вигляд S => x<u>S</u> => xyB" +
                "<br>Б) S => xS, потрібно замінити S на перший номер у таблиці, що матиме вигляд S => x<u>S</u> => xxS" +
                "<br>В) S => xS, потрібно замінити S на третій номер у таблиці, що матиме вигляд S => x<u>S</u> => xx</p>",
                2));
            //4
            this.addTest(new testData(umova + "Виходячи з прикладу S => x<u>S</u> => xxS та  умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити? Оберіть правильний варіант:" +
                "<p style='text-align:left;'>" +
                "<br>А) S => x<u>S</u>=> xxS, потрібно замінити S на перший номер у таблиці, що матиме вигляд S => xS => xx<u>S</u> => xxxS" +
                "<br>Б) S => x<u>S</u>=> xxS, потрібно замінити S на другий номер у таблиці, що матиме вигляд S => xS => xx<u>S</u> => xxyB" +
                "<br>В) S => x<u>S</u>=> xxS, потрібно замінити S на третій номер у таблиці, що матиме вигляд S => xS => xx<u>S</u> => xxx.</p>",
                2));
            //5
            this.addTest(new testData(umova + "Виходячи з прикладу S => xS => xxS => xxyB та  умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити? Оберіть правильний варіант:" +
```

```
                "<p style='text-align:left;'>" +
                "<br>А) S => xS => xx<u>S</u> => xxyB, потрібно замінити B на шостий номер у таблиці, що матиме вигляд S => xS => xxS => xxyB => xxyy" +
                "<br>Б) S => xS => xx<u>S</u> => xxyB, потрібно замінити B на п’ятий номер у таблиці, що матиме вигляд S => xS => xxS => xxyB => xxyyB" +
                "<br>В) Подальшого розв’язку даного виразу не існує.</p>",
                2));
            //6
            this.addTest(new testData(umova + "Виходячи з прикладу S => xS => xxS => xxy<u>B</u> => xxyyB  та  умови вище наведеної задачі, яку літеру потрібно обрати, та на який номер у таблиці її потрібно замінити? Оберіть правильний варіант:" +
                "<p style='text-align:left;'>" +
                "<br>А) S => xS => xxS => xxyB => xxyyB, потрібно замінити B на п’ятий номер у таблиці, що матиме вигляд S => xS => xxS => xxyB => xxyyyB" +
                "<br>Б) S => xS => xxS => xxyB => xxyyB, потрібно замінити B на шостий номер у таблиці, що матиме вигляд S => xS => xxS => xxyB => xxyyy" +
                "<br>В) Подальшого розв’язку даного виразу не існує.</p>",
                2));
        }
    }
```

Ці класи є нащадками базового класу «testOne», який задає основну поведінку:

```
public class testOne
{
    private List<testData> tests = new List<testData>();
    private int curTest = 0;

    public testData startTest() { curTest = 0; return tests[curTest]; }

    public testData nextTest()
    {
        curTest++;
        if (curTest < tests.Count)
        {
            return tests[curTest];
        }else { return null; }
    }
```

```
    public void addTest(testData data) { tests.Add(data); }
}
```

Для зберігання даних про окремий тест створено клас «testData»:

```
public class testData
{
    private byte trAn;

    public string text { get; }

    public bool tr(byte n)
    {
        if (n == trAn) { return true; }
        else { return false; }
    }

    public testData(string _text, byte _trNumber)
    {
        text = _text;
        trAn = _trNumber;
    }
}
```

Головна форма програми має назву «mainForm»:

```
public partial class mainForm : Form
{
    testOne[] tests;
    public mainForm()
    {
        tests = new testOne[2];
        tests[0] = new test1();
        tests[1] = new test2();
        InitializeComponent();
        mainForm_Resize(this, null);
    }


    private void button1_Click(object sender, EventArgs e)
    {
        this.Hide();
        int testing = (new testingChoose()).ShowForm(this);
        switch (testing)
        {
            case 1://тестування №1
```

```
                (new testingForm()).ShowForm(this, "Тренажер з теми \"Способи задання мов\": тестування №1", tests[0]);
                break;
            case 2://тестування №2
                (new testingForm()).ShowForm(this, "Тренажер з теми \"Способи задання мов\": тестування №2", tests[1]);
                break;
            default:
                MessageBox.Show("Нічого не обрано!");
                this.Show();
                break;
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        (new teoriya()).ShowDialog();
    }

    private void mainForm_Resize(object sender, EventArgs e)
    {
        button2.Width = (this.ClientSize.Width - 3 * button2.Left) / 2;
        button1.Width = button2.Width;
        button1.Left = button2.Left * 2 + button2.Width;
    }
}
```

Вона зберігає об’єкти які неохідні для початку тестування та використовує форми «teoriya» для відображення теоретичної інформації та «testingChoose» для вибору задачі для тестування. Їхній код:

```
public partial class teoriya : Form
{
    public teoriya()
    {
        InitializeComponent();
    }

    string path;

    private void InformForm_Load(object sender, EventArgs e)
    {
        path = Path.GetTempPath() + "res1_1234_21341_4234.png";
```

```
            Properties.Resources.r1.Save(path);
            webBrowser1.Navigate("about:blank");
            if (webBrowser1.Document != null) { webBrowser1.Document.Write(string.Empty); }
            webBrowser1.DocumentText = Properties.Resources.teor.Replace("!!!!!!Рисунок 1. Ієрархія Хомського для граматик!!!!!!", "<img src='" + path + "'>");
        }

        ~teoriya()
        {
            File.Delete(path);
        }
    }

    public partial class testingChoose : Form
    {
        private int state = 0;
        public testingChoose()
        {
            InitializeComponent();
        }

        public int ShowForm(Form formCall)
        {
            this.ShowDialog();
            return state;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            state = 1;
            this.Close();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            state = 2;
            this.Close();
        }
    }
```

Після вибору тестування відбувається передача відповідного об’єкту з тестами до форми «testingForm» та її відображення:

```
public partial class testingForm : Form
{
   Form parent;

   testOne testings;
   testData curTest;

   public testingForm()
   {
      InitializeComponent();

   }

   int testTrue = 0;
   int testAll = 0;
   bool testCurTrue = true;

   public void ShowForm(Form form, string title, testOne test_)
   {
      parent = form; this.Text = title; testings = test_;

      curTest = testings.startTest();
      testingForm_Resize(this, null);
      this.Show();
   }

   private void testingForm_Resize(object sender, EventArgs e)
   {
      button1.Width = button2.Width = button3.Width =
(this.ClientSize.Width - 4*10)/3;
      button1.Left = 10;
      button2.Left = button1.Left + button1.Width + 10;
      button3.Left = button2.Left + button1.Width + 10;

      webBrowser1.Left = webBrowser1.Top = 0;
      webBrowser1.Width = this.ClientSize.Width;
      webBrowser1.Height = button1.Top - 10;

      setTextInBrawser();
   }

   public void setTextInBrawser()
   {
      if (curTest != null) { setTextInBrawser(curTest.text); }
```

```
            else { this.Close(); }
        }
        public void setTextInBrawser(string ask)
        {
            bool empty = true;
            if (webBrowser1.Document != null)
            {
                HtmlElement Hask = webBrowser1.Document.GetElementById("ask");
                if (Hask != null) { Hask.InnerHtml = ask; empty = false; }
            }

            if (empty)
            {
                string s = "<html><head><style>" +
                           "body, table{font: " + 20 + "pt Arial;}" +
                           "table{ border-collapse: collapse; }  " +
                           ".b td{ border: 1px solid black;  }  " +
                           "html, body { height: 100%; } " +
                           "#main {width: 100%; height: 100%;}" +
                           "#ask {height: 100%; text-align: center;}" +
                           "p {text-align:left; margin:10px;}" +
                       "</style></head>" +
                   "<body>" +
                      "<table id='main'>" +
                         "<tr> <td colspan='3' id='ask'>" + ask + "</tr>" +
                         "<tr>  <td colspan='3' style='border-bottom: 1px solid black;'>Варіанти відповіді:</tr>" +
                      "</table>" +
                   "</body></html>";
                webBrowser1.Navigate("about:blank");
                if (webBrowser1.Document != null) { webBrowser1.Document.Write(string.Empty); }
                webBrowser1.DocumentText = s;
            }
        }
        private void ShowTrueMess(byte an)
        {
            if (!curTest.tr(an))
            { testCurTrue = false; MessageBox.Show("Відповідь не вірна, оберіть інший варіант!!!", "Обрана хибна відповідь"); }
            else
            {
                if (testCurTrue) { testTrue++; }
```

```
            testCurTrue = true;
            testAll++;
            curTest = testings.nextTest();
            setTextInBrawser();
        }
    }

    private void button1_Click(object sender, EventArgs e) { ShowTrueMess(1); }

    private void button2_Click(object sender, EventArgs e) { ShowTrueMess(2); }

    private void button3_Click(object sender, EventArgs e) { ShowTrueMess(3); }

    private void testingForm_FormClosed(object sender, FormClosedEventArgs e)
    {
        if (parent != null && !parent.IsDisposed)
        {
            (new testingEnd(testTrue, testAll)).ShowDialog();
            parent.Show();
        }
    }
}
```

По завершенню роботи форми «testingForm» відбувається передача результату тестування до форми «testingEnd» та відображення повідомлення про завершене тестування:

```
public partial class testingEnd : Form
{
    public testingEnd(int a, int b)
    {
        InitializeComponent();
        label2.Text = "Ви відповіли вірно на " + a + " питань із " + b + ".";
    }

    private void button1_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
```

Після закриття форми «testingEnd» відображається головна форма «mainForm».

ВИСНОВКИ

Мова програмування – дуже важлива складова ланка в роботі будь-якого комп’ютера.

Формальні мови і граматики - це дуже потужний математичний інструмент, який використовується в математичній та комп'ютерній лінгвістиці, описі мов програмування, розробці компіляторів в теорії програмування. Сама по собі концепція формальних граматик доволі гнучка, тому не дивно, що з'явилося багато інших інструментів, що розширюють використання та потужність граматик.

Відповідно до мети дипломної роботи були описані способи задання мов. Для задання мов використовують певні інструменти, які були описані і наведені приклади в практичній частині даної дипломної роботи.

Створено програму – тренажер на базі мови C#. За допомогою Microsoft visual studio. Програма являє собою тест, алгоритм у якої банально простий. У додатку А наведені скріншоти то як даний тренажер створювався, у додатку Б знаходиться виконавчий код тренажера. Завдяки тренажеру студент або інша особа може перевірити свої знання з даної нам теми.

У теоретичній частині описано і схематизовано усі види задання мов їх типи, поняття.

Описані основні поняття по даній темі:

- Формальні мови та алгоритми
- Регулярні та контекстовільні мови
- Формальна граматика мов програмування
- Класифікація мов програмування
- Історія мов програмування

Всі вимоги, відповідно до написання дипломної та в описані поставленої задачі, були виконані. Правила оформлення дотриманні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ємець О. О. Методичні рекомендації щодо оформлення пояснювальних записок до курсових проектів (робіт) для студентів напряму підготовки 6.040302 «Інформатика» та спеціальності 7.04030203, 8.04030203 «Соціальна інформатика» ПУЕТ / О. О. Ємець, Ол-ра О. Ємець. – Полтава : ПУЕТ, 2013. – 60 с.
2. Козлакова Г. О. Використання засобів Інтернет у науково-педагонічних дослідженнях. Нові інформаційні технології в начальних закладах України / Г. О.Козлакова// Матеріали Міжнародної конференції
3. Закон України про використання інформації [Електронний ресурс]// Законодавство України. http://zakon4.rada.gov.ua/laws/show/2657-12
4. Андрианова Г.А. Методика организации обучающей дистанционной конференции, EIDOS-LIST. - 1999. - Вып.6 (10). -).
5. Гладир А.І. Системи дистанційного навчання – огляд програмних платформ / А.І.Гладир, Зачепа Н.В., Мотруніч О.О // Проблеми вищої школи. Інновації в освіті та виробництві. Комп'ютерні технології в освіті та виробництві. – Кременчук : КНУ ім. М. Остроградського. – с. 43-44.
6. Синепол В.С., Цикин И.А. Применение современных программно-аппаратных средств компьютерной видео конференцсвязи для создания интерактивных информационно-обучающих систем. Всероссийская научно-методическая конференция "Телематика 95", С.-Петербург, 1995
7. Офіційна презентація продукту AdobeAcrobatConnect в Україні [Електронний ресурс]. – Режим доступу: http://www.pre-paid.com.ua/2434.html. – 26 березня 2009 р.
8. Игошин В. И. Математическая логика и теория алгоритмов : учеб. пособие для студ. высш. учеб. заведений / В. И. Игошин. — 2–е изд., стер. — М. : Издательский центр «Академия», 2008. — 448 с.

9. Игошин В. И. Задачи и упражнения по математической логике : учеб. пособие для студ. высш. учеб. заведений / В. И. Игошин. — 2–е изд., стер. — М. : Издательский центр «Академия», 2008. — 448 с.
10. C# [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/C_Sharp.
11. Герман О.В. Программирование на Java и C# для студента / О.В. Герман, Ю.О. Герман. – СПб. : БХВ-Петербург, 2005. – 512 с.