

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики



Реалізація бази знань за допомогою системи PROTEGE

Текстова частина до курсової роботи

за спеціальністю «Інженерія програмного забезпечення»- 121

Керівник курсової роботи

Зав. кафедри мультимедійних
систем, доцент, к.ф.-м.н.

Жежерун О.П.

(підпис)

“ _____ ” _____ 2020 р.

Виконав студент ІІІЗ-3:

Пархоменко Д.О.

“ _____ ” _____ 2020 р.

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра мережних технологій факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри мультимедійних
систем, доцент, к.ф.-м.н.

_____ О.П. Жежерун

„____” _____ 2020 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Пархоменку Данилу Олександровичу факультету Інформатики
третього курсу

ТЕМА: Реалізація бази знань за допомогою системи PROTEGE

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1. Огляд задач шкільної програми з геометрії.
2. Розробка схеми алгоритму вирішення задач з планіметрії
3. Розробка програми зчитування умови задачі та її розв'язання.

Висновки

Список літератури

Додатки (за необхідністю)

Дата видачі “____” _____ 2020 р. Керівник _____
(підпис)

Завдання отримав _____
(підпис)

Тема: Реалізація бази знань за допомогою системи PROTEGE

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу
1.	Отримання завдання на курсову роботу	10.12.2019
2.	Вивчення прикладів задач з геометрії	30.01.2020
3.	Виконати аналіз сучасних методів розв'язання подібних задач	04.02.2020
4.	Розробка алгоритму зчитування задач науковою мовою і їх розв'язання	10.02.2020
5.	Вивчення правил розв'язання задач з геометрії	25.02.2020
6.	Створення онтології за допомогою Protégé	05.03.2020
7.	Створення ієрархії на основі онтології в Java	20.03.2020
8.	Спроби вирішення задач з геометрії за допомогою мови Java	05.04.2020
9.	Написання текстової частини курсової роботи	30.04.2020
10.	Узгодження курсової роботи з науковим керівником	05.05.2020
11.	Написання презентації для захисту курсової роботи	10.05.2020
12.	Захист курсової роботи	18.05.2020

Студент Пархоменко Данило Олександрович

Керівник Жежерун Олександр Петрович

“ _____ ”

Зміст

Анотація.....	5
Вступ	6
РОЗДІЛ 1: Аналіз задач з планіметрії шкільної програми.....	8
1.1 Загальна концепція вирішення шкільних задач.....	8
1.2 Аналіз тексту задач	12
РОЗДІЛ 2: Аналіз онтології геометричних фігур	17
2.1 Загальна інформація про онтології.....	17
2.2 Опис реалізованої онтології геометричних фігур	22
РОЗДІЛ 3: Реалізація алгоритму вирішення геометричних задач.....	24
3.1 Загальний опис алгоритму	24
3.2 Аналіз практичної частини вирішення задач.....	27
3.3 Реалізація алгоритму	32
3.4 Огляд структури програми	36
Список літератури	39
Додаток А. Повідомлення про не вміння програми вирішувати введену задачу.....	41
Додаток Б. Вирішена задача на тему “трикутники”	42
Додаток В. Вирішена задача на тему “паралельні прямі” з формулюванням «в 4 рази більший»	43
Додаток Г. Вирішена задача на тему “паралельні прямі” з формулюванням «на 20 градусів менший»	44
Додаток І. Натиснута кнопка з порожнім полем для вводу задачі	45

Анотація

У даній курсовій роботі наведено загальні принципи вирішення різних задач. Проаналізовано тип мови в математичних задачах та принцип семантичного аналізу цих задач. Продемонстровано онтологію геометричних фігур, яка дає змогу зрозуміти зв'язки між фігурами та принципи їх взаємодії. Також в роботі наведено алгоритм вирішення геометричних задач з шкільної програми та описані труднощі пов'язані з парсингом штучної мови цих задач.

В роботі використовується система Protégé 5.5.0, Java 1.8, Java Swing.

Ключові слова: онтологія, геометрія, задачі, семантичний аналіз, редактор онтологій, природна мова, штучна мова, Protégé, Java, Java Swing, PullEnti.

Вступ

Розв’язування задач шкільної програми геометрії досить цікава та неординарна проблема, що потребує гарно розвинутого просторового мислення та уважності. Геометрія складається з багатьох розділів, починаючи з планіметрії, що вивчає двовимірні фігури (точка, лінія, коло, трикутник і так далі), закінчуючи стереометрією, що вивчає тривимірні фігури (куб, сфера, піраміда та інші).

Однак, на сьогоднішній день не має певного інструменту розв’язання задач з геометрії, написаних простим, зрозумілим для людей, текстом. Звісно в інтернеті є багато калькуляторів, які дозволяють знайти певну величину (периметр, площу), приймаючи на вхід готові значення, з яких складаються ці величини^{[1][2]}. В той же час, в інтернеті немає ні одного сайту чи програмного застосунку, який би дозволив надати на вхід умову задачі і в результаті видати готове рішення. Зазвичай в інтернеті такі задачі вирішуються людьми, а не програмним застосунком. Мета курсової роботи – дослідити та з’ясувати, які дії треба зробити для створення застосунку здатного аналізувати природню мову задачі та видавати потрібний результат. Джерелом дослідження виступає підручник з геометрії за сьомий клас авторства Мерзляка А.Г «Геометрія».

Робота складається з трьох розділів.

Перший розділ присвячено вивченню та дослідженню задач з шкільного курсу геометрії. Наведено конкретні приклади схожості задач між собою, а також їх розбіжності. Проаналізовано тривіальні способи вирішення цих задач та як ці способи можна втілити в нашому програмному застосунку.

В другій частині розглянуто побудовану базу знань за допомогою системи Protégé та до яких висновків по реалізації можна прийти, проаналізувавши отриману базу знань.

Третю частину присвячено огляду ієрархії класів в Java на основі бази знань з Protégé. Розглянуто певні проблеми пов'язані з написанням програмного застосунку, для розв'язання задач з геометрії.

Постановка задачі

1. Виконати аналіз задач з шкільної програми геометрії та з'ясувати спосіб вирішення подібних задач.
2. Зробити базу знань з геометрії за допомогою Protégé.
3. Розробити алгоритм вирішення задач за допомогою програмного коду.

Аналіз задач з планіметрії шкільної програми

Загальна концепція вирішення шкільних задач

Для аналізу задач було обрано підручник з геометрії за сьомий клас авторства Мерзляка А.Г «Геометрія».^[3] Саме з цього підручника учні починають шлях у науку під назвою геометрія. Спочатку школярі вивчають фундаментальні основи геометрії, а саме поняття точки та прямої. Але загалом у підручнику є три основні теми: трикутники, коло, паралельні прямі. Для вирішення будь-якої задачі треба знати певні правила її вирішення. Візьмемо для прикладу задачу з теми «трикутники».

Периметр рівнобедренного трикутника дорівнює 28 см, а бічна сторона - 10 см. Знайдіть основу трикутника.

Для вирішення даної задачі учню треба володіти наступною інформацією:

1. Що таке «периметр». Периметр – це сума всіх сторін трикутника
2. Що таке «рівнобедренний трикутник». Трикутник, який складається з двох бічних сторін та основи
3. Що таке «бічна сторона рівнобедреного трикутника». Бічні сторони трикутника рівні
4. Що таке «основа рівнобедреного трикутника».

Знаючи відповіді на всі ці питання, учень може легко розв'язати наступну задачу:

$$28=10+10+x$$

$$x = 28-20$$

$$x = 8 \text{ см.}$$

Задачу розв'язано. Які висновки для написання програми можна зробити з цього прикладу? Програма для вирішення задач з геометрії має знати спосіб вирішення подібних задач, а для цього треба знати всі складові задачі та всі правила за якими вони вирішуються. В геометрії «правилами» називають аксіоми та теореми, а також наслідки з теорем.

«Аксіо́ма (грец. ахі́о́та — загальноприйняте, безперечне, від ахіо — вважаю гідним, наполягаю, вимагаю) — вихідне положення, самоочевидний принцип; у переносному значенні те, що не потребує жодних доведень; твердження, заперечення якого заперечує основи логічного мислення.»^[a]

«Теоре́ма (грец. θεώρημα — «вигляд, уявлення, положення») — твердження у математиці, для якого в теорії, що розглядається, існує доказ (інакше кажучи, доведення). Вихідним пунктом для теорем є аксіоми, які приймаються істинними без всяких доказів або обґрунтувань.»^[b]

Наслідки – менш важливі твердження-теореми. (Тобто це теж теореми, але які мають не таку важливість). Проте не можна позбавляти уваги наслідки, як необхідних структурний елемент у розв'язанні задачі учнем або програмою.

^a Вікіцитати. Аксіома.

<https://uk.wikiquote.org/wiki/%D0%90%D0%BA%D1%81%D1%96%D0%BE%D0%BC%D0%B0>.

^b Вікіпедія. Теорема

<https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0>.

Подивимось на конкретний приклад теорем, аксіом та наслідків для теми з підручника. Наприклад, для теми «паралельні прямі» список всіх правил виглядає так:

1. Дві прямі називаються **паралельними** якщо вони не перетинаються.
2. Дві прямі, які перпендикулярні до третьої прямої - **паралельні**.
3. Через точку, яка не лежить на даній прямій, проходить тільки одна пряма, паралельна даній
4. Якщо дві прямі паралельні третій прямій, то вони всі паралельні.
5. Якщо дві прямі перетнуті третьою прямою, то ця пряма називається **січною**.
6. Кути, які знаходяться між двох прямих, які перетинає січна називаються **внутрішніми**.
7. Кути, які знаходяться ззовні двох прямих, які перетинає січна називаються **зовнішніми**.
8. Кути, які є внутрішніми і знаходяться на одній стороні від січної називаються **внутрішніми односторонніми**.
9. Кути, які є внутрішніми і знаходяться на різних сторонах від січної називаються **внутрішніми різносторонніми**.
10. Кути, які є зовнішніми і знаходяться на одній стороні від січної називаються **зовнішніми односторонніми**.
11. Кути, які є зовнішніми і знаходяться на різних сторонах від січної називаються **зовнішніми різносторонніми**.
12. Кути, які знаходяться на одній стороні від січної біля різних прямих і один з них внутрішній, а один зовнішній називаються **відповідними**.
13. Якщо різносторонні кути, утворенні при перетині двох прямих січною, рівні, то прямі паралельні (також правильне протилежне твердження від паралельних прямих).
14. Якщо сума односторонніх кутів, утворених при перетині двох прямих січною $= 180$ градусів, то прямі паралельні (також правильне протилежне твердження від паралельних прямих).

Рисунок 1.1 – Теореми, аксіоми та наслідки для теми «паралельні прямі».

Існує нюанс який потребує вирішення. Програма може розв'язувати задачі з різних тем та різного ступеня складності, проте програмі, як і учню, потрібно знати всі правила, необхідні для розв'язання конкретного типу задач, враховуючи їх особливості та рівень складності.

Важливим етапом у розв'язанні задач є правильний поділ конкретної задачі на структурні елементи, їх класифікація та зв'язок структурних елементів з аксіомами, теоремами і наслідками, як це було зазначено вище на прикладі. Оскільки інтелектуальний процес людини під час розумової праці часто виконує певні дії інтуїтивно, не аналізуючи їх, то при написанні програми, для забезпечення якісного виконання поставленої мети варто врахувати це та проаналізувати в правильному порядку логічно-послідовні дії людини та продублювати їх в автоматичному режимі програми. Порівнюючи процес розв'язання задач програмою та людиною, з обох сторін є певні недоліки та переваги, якими потрібно скористатись для оптимізації процесів. Враховуючи особливості варто також зазначити про важливу науково-теоретичну частину. А саме алгоритми кожної дії, зазначеної програмою, мають бути не тільки чітко оформлені та автоматизовані, а й аргументовані загальними математичними законами, правилами геометрії, логічним порядком та мовною лексикою, якою написана задача. Це значно спростить процес розв'язання, оптимізує його подібно до природного інтелектуального процесу, проте не буде підвладна помилкам, притаманним людині.

Для повноцінної роботи програми варто врахувати усі вище зазначені важливі елементи, а також проблеми, які можуть виникнути в процесі їх кодування або безпосередньо в роботі.

Аналіз тексту задач

Задля парсингу будь-якого тексту, включаючи текст задач, спочатку необхідно визначити до якої мови належить текст задач з геометрії (та і загалом будь-яких задач з математики).

Існує два основних типи мов *природні* та *штучні*.

Природні мови - це історично сформовані в суспільстві звукові (мова), а потім і графічні (лист) інформаційні знакові системи. Природні мови виникли для закріплення і передачі накопиченої інформації в процесі спілкування груп людей або двох людей. Природні мови виступають носіями багатовікової культури людства і відрізняються багатими виразними можливостями, а також охоплюють найрізноманітніші сфери життя.

В силу певних особливостей природних мов, їх не завжди вдається використовувати в процесі наукового пізнання. Виділяють такі особливості

1) **багатозначність** - багато слів і мовні вирази природної мови можуть приймати різні значення в залежності від контексту, це пов'язано з омонімією, наприклад, слова "світ", "коса", "рукав" та інші;

2) **некомпозиційність**, тобто відсутність в природній мові правил, за допомогою яких можна було б визначити точне значення складного висловлювання, не знаючи контекст висловлювання, хоча значення всіх вхідних в нього слів відомі. Наприклад, фраза "Він довго сідав на коня зі зламанною ногою" може тлумачитись двояко: а) Вершник був зі зламанною ногою; б) Кінь був зі зламанною ногою;

3) **самовикористання**, тобто коли словосполучення говорять самі за себе. Наприклад, "Я брешу".

Штучні (наукові) мови створюються спеціально для вирішення певних завдань пізнання. Вони з'явилися як формалізовані мови науки - математики,

фізики, хімії, програмування. Штучні мови - це допоміжні знакові системи, що створюються на базі природних мов для точної і економної передачі наукової та іншої інформації. Вони конструюються за допомогою природної мови або раніше побудованого штучної мови.

Наукові мови підкоряються деяким нормативним принципам нормативним принципам: однозначності, предметності і взаємозамінності.

Відповідно до принципу **однозначності** вираз, що використовується в якості імені, має бути ім'ям тільки одного предмета, якщо це одиничне ім'я, а якщо це загальна назва, то цей вислів має бути ім'ям, загальним для всіх предметів одного класу. У природній мові цей принцип не завжди дотримується, але його необхідно дотримуватися при побудові штучних мов, наприклад мови логіки предикатів.

Принцип однозначності виключає омонімію, тобто позначення одним словом різних об'єктів, що часто зустрічається в природних мовах (наприклад, слово "коса" може означати і вид зачіски, і сільськогосподарське знаряддя праці, і піщану мілину).

Відповідно до принципу **предметності** у висловлюваннях має затверджуватися або заперечуватися щось про значення імен, що входять в речення, а не щось про самі імена. Слід, звичайно, мати на увазі, що значеннями деяких імен є самі імена. Такі випадки не суперечать принципу предметності. Наприклад, у реченні "Матерія первинна, а свідомість, то інше" слово "матерія" - це ім'я об'єктивної реальності, а в реченні "" Матерія "- це філософська категорія" слово "матерія", взяте в лапки, - це ім'я імені, ім'я категорії. Такі імена називаються *лапковими* іменами. Іноді в природній мові зустрічаються випадки, коли ім'ям імені є саме вихідне ім'я. Наприклад, у реченні "Слово" стіл "складається з чотирьох букв" слово "стіл" є ім'ям самого цього слова. Таке вживання імен, коли слова позначають самі себе,

називається автономним. Автономне вживання виразів неприпустимо в наукових мовах, оскільки воно призводить до непорозумінь.

Для вказівки на автономне вживання виразів використовуються курсив або лапки. Змішання звичайного і автонімного вживання виразів веде до логічних помилок в міркуваннях. Прикладом такої помилки може служити наступне міркування: "Собака гризе кістку." Собака "- іменник. Отже, іменник гризе кістку".

Принцип **взаємозамінності**: якщо в складному імені замінити частину, в свою чергу є ім'ям, іншим ім'ям з тим же значенням, то значення, отримане в результаті такої заміни складного імені, має бути тим же, що і значення вихідного складного імені. Наприклад, у реченні "Аристотель навчав Олександра Македонського філософії", слово "Аристотель" можна замінити словами "творець силогістики".^[4]

Проаналізуємо текст в задачах з підручника з геометрії за сьомий клас авторства Мерзляка А.Г «Геометрія». Для прикладу візьмемо задачу з теми «паралельні прямі»:

Один з односторонніх кутів, утворених при перетині двох паралельних прямих січною, у 4 рази більший за другий. Знайдіть ці кути.

В цій задачі немає багатозначності, некомпозиційності та самовикористання. З цього можна зробити висновок, що текст задачі не відноситься до природної мови. Визначимо чи відноситься цей текст до штучної мови.

Принцип однозначності виконується, бо є певні імена у частин задачі, такі як «паралельні прямі», «січна», «кути», «односторонні кути». Тобто в геометрії нема декількох означень для цих термінів.

Принцип предметності також виконується, так як немає автономного вживання виразів в задачах.

Принцип взаємозамінності виконується, краще це продемонструвати на іншій задачі, наприклад в задачі про трикутники слово «трикутник» можна замінити на «фігура з трьома кутами».

Отже, всі принципи штучної мови виконуються. Це означає, що в теорії є можливість розпарсити текст без використання семантичного аналізатора природньої мови. Подивимось на загальний синтаксис задач з теми «трикутники»:

- 1)
 - a. Периметр
 - b. Знайдіть, Доведіть
 - i. Третю, другу. Дві інші. Сторони, кут.
 - ii. Периметр
 - c. Точки (назва) належать
 - d. У трикутнику (назва)
 - e. (Наступний крок)
- 2)
 - a. (Наступний крок)
 - b. Сторона
 - i. Основа, бічна (назва)
 - ii. (наступний крок)
 - c. Кут
 - d. Медіани, висота, бісектриси (назва)
- 3) Види
 - a. Рівнобедрений
 - b. Вертикальний, суміжний (назва). До кута (назва)
 - c. (наступний крок)
- 4) Трикутник (назва)
- 5) Дорівнює
- 6) (значення)

Рисунок 1.2 – Структура задач з теми «трикутники»

Бачимо, що навіть задачі з однієї теми мають доволі різну структуру. Слова часто міняються місцями, що робить аналіз тексту без використання певного семантичного аналізатора вкрай важким. Не варто забувати, що це тільки

структура задач з однієї теми, а в планіметрії дуже багато різних тем, задачі в яких також мають різну структуру.

Також дуже часто в задачах слова «дорівнює», «рівняється» замінюється на тире. Наприклад, задача про бічні сторони і основу з першого розділу:

Периметр рівнобедреного трикутника дорівнює 28 см, а бічна сторона - 10 см. Знайдіть основу трикутника.

В цій задачі після слів «бічна сторона» пропущений присудок «дорівнює» і в такому випадку парсинг задачі теж значно ускладнюється.

Отже, розібрати штучну мову геометричних задач без використання семантичного аналізатора не є можливим.

Аналіз онтології геометричних фігур

Загальна інформація про онтології

Для аналізу структури геометричних фігур і їх зв'язків між собою було створено онтологію за допомогою системи Protégé.

Неформально онтологія представляє собою деякий опис погляду на світ стосовно конкретної області інтересів. Цей опис складається з термінів і правил використання цих термінів, які обмежують значення цих термінів в рамках конкретної області.

На формальному рівні онтологія - це система, що складається з набору понять і набору тверджень про ці поняття, на основі яких можна описувати класи, відносини, функції та індивіди.

Одне з найвідоміших визначень онтології дав Том Грубер.

Том Грубер – це американський науковець, винахідник та підприємець, який акцентується на системах обміну знаннями і колективного розуму. Він зробив основоположну роботу в інженерії онтологій та гарно відомий за визначення онтологій у контексті штучного інтелекту. Також Том був одним з розробників Siri – розумного особистого помічника та навігатора знань. На даний момент Siri є невід'ємною частиною прошивки всіх девайсів Apple.

А саме визначення Тома звучить наступним чином: Онтологія - це точна специфікація концептуалізації.

Концептуалізація - це структура реальності, що розглядається незалежно від словника предметної області і конкретної ситуації.

Наприклад, якщо ми розглядаємо просту предметну область, що описує кубики на столі, то концептуалізацією є набір можливих

положень кубиків у просторі, а не конкретне їх розташування в даний конкретний момент часу.

Пізнішою модифікацією визначення Грубера є таке визначення:

Онтологія - це формальна специфікація узгодженої концептуалізації.

Під узгодженої концептуалізацією мається на увазі, що дана концептуалізація не є приватна думка, а є спільною для деякої групи людей.

Сформульовано ще досить багато різних визначень онтології.

Наприклад, Ніколо Гуаріно визначає онтологію наступним чином:

Онтологія - це формальна теорія, що обмежує можливі концептуалізації світу.

Деякі визначення відображають способи, якими автори будують і використовують онтології, наприклад: Онтологія - це ієрархічно структурована множина термінів, що описують предметну область, яка може бути використана як вихідна структура для бази знань.

Основними компонентами онтології можуть бути:

- класи (або поняття),
- функції,
- відносини (або властивості, атрибути),
- екземпляри (або індивіди),
- аксіоми.

Класи або поняття використовуються в широкому сенсі. Поняттям може бути будь-яка сутність, про яку може бути дана будь-яка інформація. Класи - це абстрактні групи, колекції або набори об'єктів. Вони можуть включати в себе екземпляри, інші класи, або ж поєднання і того, і іншого. Класи в онтологіях зазвичай організовані в таксономії - ієрархічну класифікацію понять по відношенню

включення. Наприклад, класи Чоловік і Жінка є підкласами класу Людина, яка в свою чергу входить в клас Ссавці.

Відносини представляють тип взаємодії між поняттями предметної області. Формально n -арні відношення визначаються як підмножина похідного n множин: $R \subseteq C_1 \times C_2 \times \dots \times C_n$. Приклад бінарного відношення - відношення ЧАСТИНА-ЦІЛЕ. Відношення теж можуть бути організовані в таксономії по включенню; наприклад, відносини бути_батьком_для і бути_матір'ю_для на безлічі людей містяться в відношенні бути_батьками_для, яке в свою чергу міститься в відношенні бути_предком_для.

Функції - це спеціальний випадок відношень, в яких n -й елемент відношення однозначно визначається $n-1$ попередніми елементами. Формально функції визначаються наступним чином: $F: C_1 \times C_2 \times \dots \times C_{n-1} \rightarrow C_n$. Прикладами функціональних відношень є відношення бути_матір'ю_для на безлічі людей, або ціна_потриманого_автомобіля, яка обчислюється залежно від моделі автомобіля, дати виготовлення і пробігу.

Аксіоми використовуються, щоб записати висловлювання, які завжди правдиві. Вони можуть бути включені в онтологію для різних цілей, наприклад, для визначення комплексних обмежень на значення атрибутів, аргументи відносин, для перевірки коректності інформації, описаної в онтології, або для виведення нової інформації.^[5]

Спільне використання людьми або програмними агентами загального розуміння структури інформації є однією з найбільш загальних цілей розробки онтологій^[6]. Наприклад, нехай, кілька різних веб-сайтів містять інформацію з медицини або надають інформацію про платні медичні послуги, які оплачуються через Інтернет. Якщо ці веб-сайти спільно використовують і публікують одну і ту ж базову онтологію

термінів, якими вони всі користуються, то комп'ютерні агенти можуть отримувати інформацію з цих різних сайтів і накопичувати її. Агенти можуть використовувати накопичену інформацію для відповідей на запити користувачів або як вхідні дані для інших додатків.

Забезпечення можливості використання знань предметної області стало однією з рушійних сил недавнього сплеску в вивченні онтологій. Наприклад, для моделей багатьох різних предметних областей необхідно сформулювати поняття часу. Це уявлення включає поняття тимчасових інтервалів, моментів часу, відносних заходів часу і т.д. Якщо одна група вчених детально розробить таку онтологію, то інші можуть просто повторно використовувати її в своїх предметних областях. Крім того, якщо нам потрібно створити велику онтологію, ми можемо інтегрувати кілька існуючих онтологій, що описують частини великої предметної області. Ми також можемо повторно використовувати основну онтологію, таку як UNSPSC (Стандартний кодекс товарів і послуг Організації Об'єднаних Націй - це систематика продуктів і послуг для використання в електронній комерції. Це чотирирівнева ієрархія, кодована як восьмизначне число, а необов'язковий п'ятий рівень додає ще дві цифри), і розширити її для опису предметної області яка нас цікавить.

Створення явних припущень в предметній області, що лежать в основі реалізації, дає можливість легко змінити ці припущення при зміні наших знань про предметну область. Жорстке кодування припущень про мир на мові програмування призводить до того, що ці припущення не тільки складно знайти і зрозуміти, але і також складно змінити, особливо не програмісту. Крім того, явні специфікації знань в предметній області корисні для нових користувачів, які повинні дізнатися значення термінів предметної області.

Відділення знань предметної області від оперативних знань - це ще один варіант загального застосування онтологій. Ми можемо описати завдання конфігурації продукту з його компонентів відповідно до необхідної специфікацією і впровадити програму, яка робить цю конфігурацію незалежною від продукту і самих компонентів^[7]. Після цього ми можемо розробити онтологію компонентів і характеристик ЕОМ і застосувати цей алгоритм для конфігурування нестандартних ЕОМ. Ми також можемо використовувати той же алгоритм для конфігурування ліфтів, якщо ми надамо йому онтологію компонентів ліфта.^[8]

Аналіз знань в предметній області можливий, коли є декларативна специфікація термінів. Формальний аналіз термінів надзвичайно цінний як при спробі повторного використання існуючих онтологій, так і при їх розширенні.^[9]

Часто онтологія предметної області сама по собі не є метою. Розробка онтології на кшталт визначенню набору даних і їх структури для використання іншими програмами. Методи вирішення завдань, доменно-незалежні програми та програмні агенти використовують в якості даних онтології та бази знань, побудовані на основі цих онтологій. Наприклад, ми розробляємо онтологію вин і їжі, а також відповідні комбінації вин і страв. Потім цю онтологію можна буде використовувати як основу для додатків в наборі інструментів для управління рестораном: Один додаток міг би складати список вин для меню на поточний день або відповідати на запити офіціантів і відвідувачів. Інша програма могла б аналізувати інвентарний перелік винного льоху і пропонувати категорії вин для поповнення і конкретні вина для закупівлі до наступним меню або для куховарських книг.

Опираючись на цю інформацію, можна проаналізувати онтологію геометричних фігур.

Опис реалізованої онтології геометричних фігур

Для аналізу структури геометричних фігур, розуміння як ті чи інші геометричні елементи пов'язані між собою було створено онтологію за допомогою системи Protege.

Спочатку планувалось взяти готову онтологію з відкритих бібліотек, наприклад, Ontolingua – найбільша відкрита бібліотека онтологій. Але Стендфордський університет закрив проект Ontolingua та електронний ресурс, який надавав доступ до самої бібліотеки. Тому довелось створювати онтологію власноруч.

Основою онтології є класи Точка та Пряма – підкласи класу Thing. Від Точки та Прямої наслідуються клас Відрізок. Від Відрізка наслідуються багато різних геометричних елементів, таких як Медіана, Бісектриса, Висота, Хорда, Радіус, Діаметр, а головне, що з відрізків складаються геометричні фігури, тому від відрізка наслідуються клас Геометрична_Фігура, від якої вже наслідуються Трикутник, Чотирикутник та Багатокутник.

Також є різні типи трикутників: Прямокутний, Рівнобедрений, Рівносторонній, Тупокутний, та чотирикутників: Трапеція, Прямокутній, Квадрат, Паралелограм.

Окремо винесені Паралельні_прямі, як окрема тема геометрії.

Реалізована онтологія у вигляді графу виглядає наступний чином:

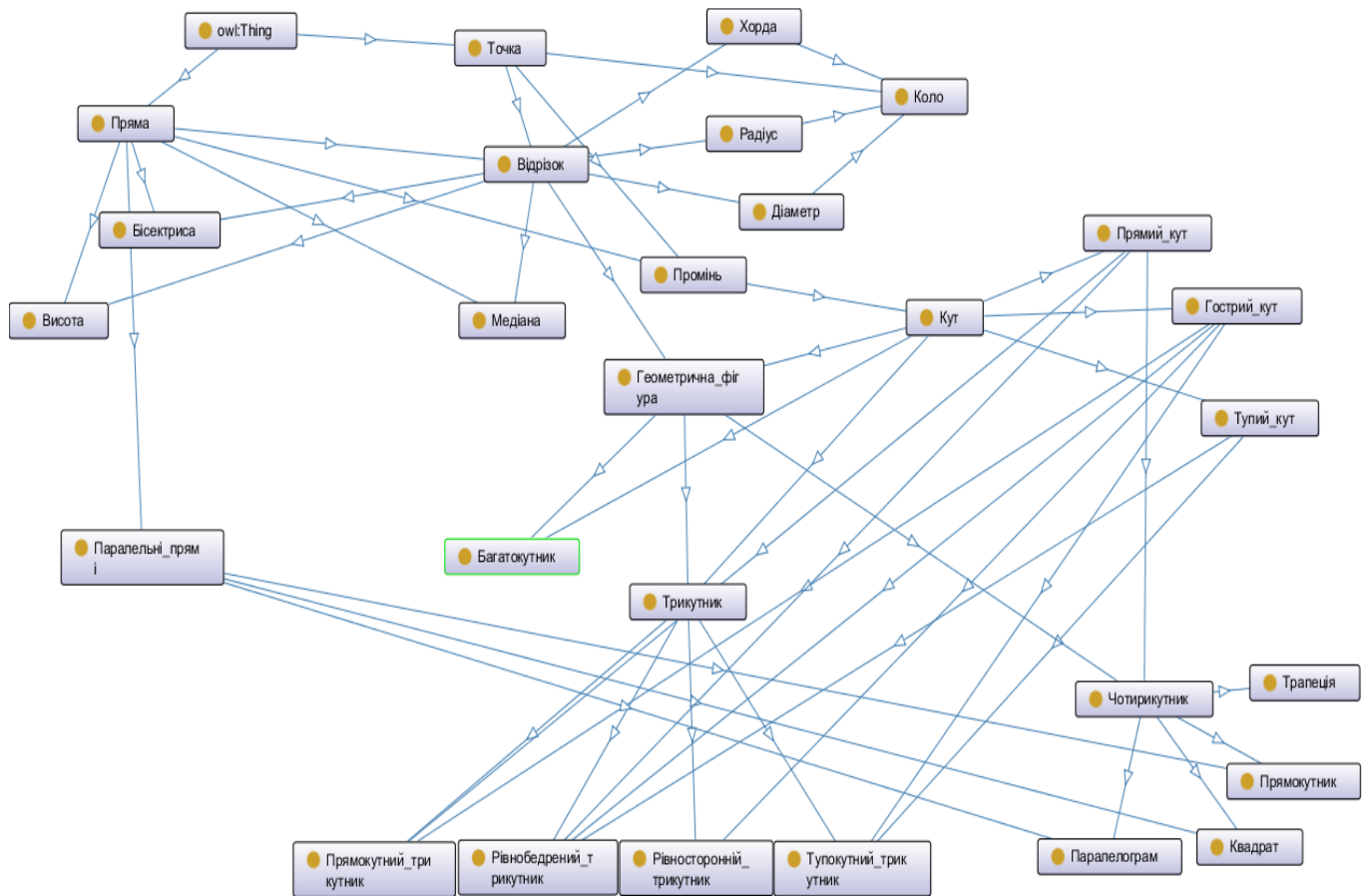


Рисунок 2.1 – Онтологія геометричних фігур у вигляді графу.

Реалізована онтологія виглядає дещо громіздко, але дозволяє зрозуміти яка приблизна структура ієрархії Java класів для розв’язання задач з геометрії має вийти в результаті.

Реалізація алгоритму вирішення геометричних задач

Загальний опис алгоритму

Для того, щоб застосунок зміг вирішити задачу з геометрії йому треба знати певний послідовний набір дій. В програмуванні такий набір дій називають алгоритмом.

“Алгоритм — це точний і зрозумілий опис послідовності дій над заданими об’єктами, що дозволяє одержати кінцевий результат.

Складання алгоритму починається з розбивання описуваного процесу на послідовність окремих кроків. Властивість розбивання алгоритму на окремі кроки називають дискретністю алгоритму. Кожний крок алгоритму формулюється у вигляді чітких інструкцій, тобто визначених команд виконавцю. Наприклад, указати послідовність дій, які необхідно виконати для обчислення виразу $A * X = B$.”

Гарний алгоритм має підпорядковуватись деяким обов’язковим властивостям:

1. **Скінченність.** Виконання кожного алгоритму повинно завершуватися за скінченну кількість кроків.
2. **Результативність.** Виконання алгоритму завжди повинно приводити до певного результату. Виконання не може закінчуватися невизначеною ситуацією або ж не закінчуватися взагалі.
3. **Формальність.** Виконавець відповідно до алгоритму повинен одержати результат, не вникаючи в його суть. Ця властивість має особливе значення для автоматизованого виконання алгоритмів.

4. Визначеність. Будь-який алгоритм потрібно описати так, щоб під час його виконання у виконавця не виникало двозначних указівок. Тобто різні виконавці згідно з алгоритмом повинні діяти однаково та одержати один і той самий результат.
5. Масовість. За допомогою створеного алгоритму можна розв'язувати цілий клас задач.
6. Зрозумілість. В алгоритмі повинні бути лише ті вказівки, які знайомі виконавцеві.^[10]

Розглянемо алгоритм вирішення задач з геометрії, який має бути в застосунку:

1. Зчитати текст задачі.
2. Розпарсити текст задачі за допомогою семантичного аналізатора української мови.
3. Визначити, які дані нам надаються в умові задачі.
 - a. Визначити, яка фігура, або сукупність фігур надані в задачі.
 - b. Визначити, які величини відомі з умови задачі.
4. Визначити що саме потребується знайти. Іншими словами, зрозуміти, який результат має вийти після вирішення задачі.
5. Створити об'єкт класу елемента, який надається в умові задачі з параметрами, які є в задачі. Наприклад, трикутник з відомою основою, коло з відомим радіусом, чотирикутник з стороною та двома кутами, паралельні прямі тощо.
6. Отримати потрібний параметр з класу геометричного елемента, який треба було знайти.
7. Вивести результат.

Реалізувавши даний алгоритм в коді, програма буде здатна вирішувати більшість задач з геометрії. Звісно в підручнику з геометрії є ще багато задач на доведення або на рисування певної

фігури, але в цій курсовій роботі були розглянуті тільки задачі на знаходження певного результату у вигляді числа.

Розберемо, як реалізувати даний алгоритм вже безпосередньо в програмі.

Аналіз практичної частини вирішення задач

Першим кроком в алгоритмі потребується зчитати текст задачі. Це легко робиться за допомогою простого інтерфейсу користувача. В даній роботі в якості інтерфейсу користувача використовувалась бібліотека Java Swing. Головне вікно програми виглядає наступним чином:

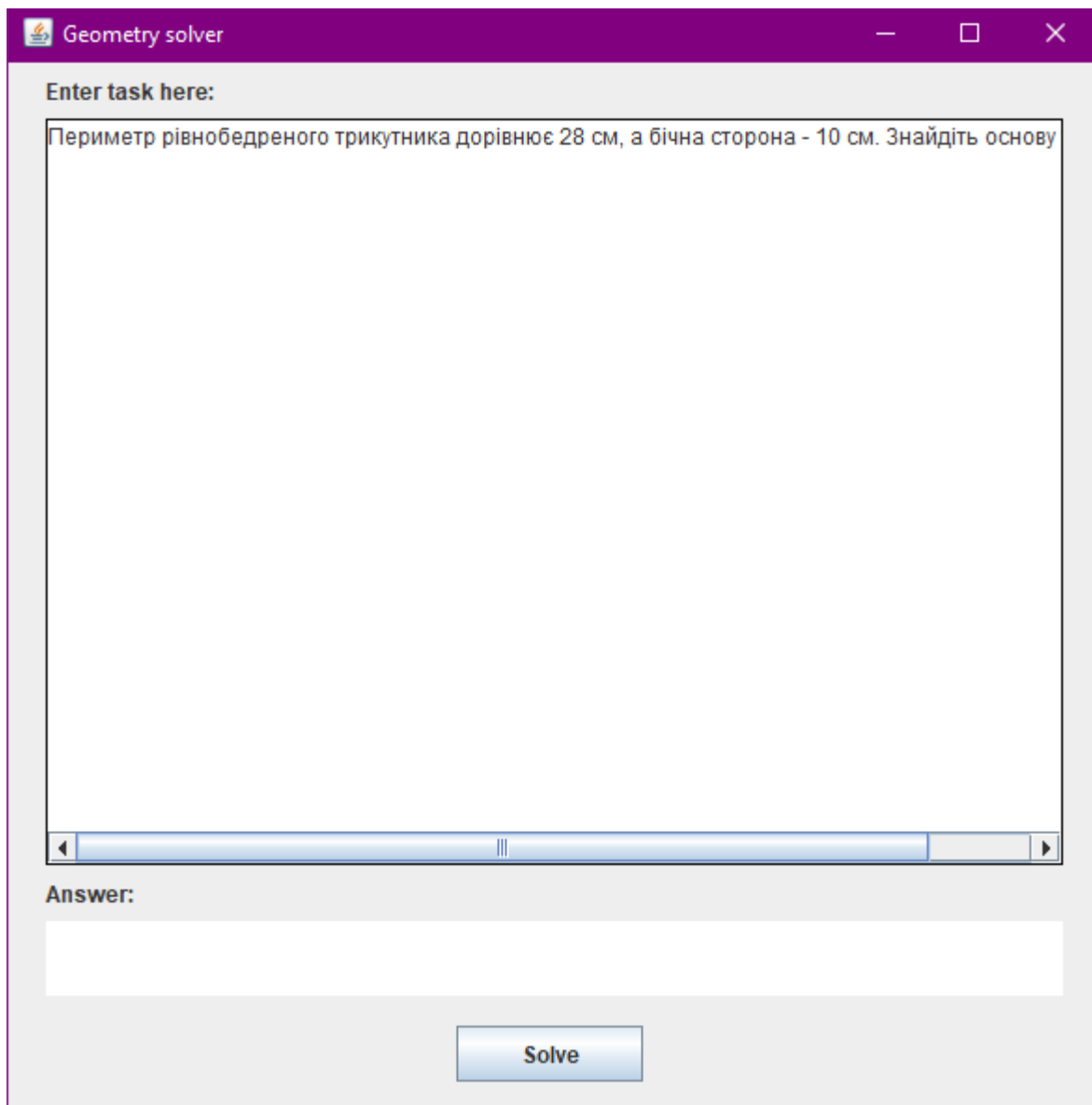


Рисунок 3.1 – Головне вікно програми

Інтерфейс виглядає дещо просто, але для зі своєю задачею, а саме зчитування даних про задачу та виведення результату задачі, інтерфейс справляється.

Для спроби семантичного аналізу було використано бібліотеку PullEnti.

SDK Pullenti (Puller of Entities) призначене для вирішення завдання аналізу тексту і виділення іменованих сутностей з неструктурованих російськомовних та українськомовних текстів в рамках інформаційних систем, що розробляються на .NET Framework 2.0 (і вище), .NET Core 2.0 (і вище), Java, JavaScript (Node.js) і Python 3.^[11]

SDK складається із загальної та спеціалізованої частини. Загальна частина містить реалізацію спільних алгоритмів лінгвістичного аналізу, а також підтримку моделі даних. Спеціалізована частина складається з так званих аналізаторів, що реалізують виділення сутностей певних типів (персони, організації та ін.).

Іменована сутність - це об'єкт, який містить набір значень атрибутів, який відрізняє його від інших об'єктів цього ж типу. Звичайно, це не суворе визначення. Кожен аналізатор сам вирішує, що вважати сутністю, а що ні. Наприклад, у фразі «Вася пішов гуляти» персону НЕ виділиться стандартним аналізатором «Персони», так як тільки по одному імені або по прізвищу персону не виділяється щоб уникнути втрати точності. Хоча якщо в тексті персону з ім'ям «Вася» в іншому місці буде виділена (наприклад, з фрази «Вася Іванов – хороший хлопчик »), то і в інших місцях вона буде виділятися по одному імені (якщо, звичайно, в тексті немає інших «Вась»). Однак інший аналізатор може вести себе зовсім по-іншому по відношенню до «Васі».

Бібліотека чудово працює на виділення іменованих сутностей, але семантичний аналіз тільки знаходиться в стадії розробки. Тому з використанням даної бібліотеки є певні проблеми.

По-перше, аналізатор не може визначити зв'язок між підметом і додатком якщо пропущений присудок (*Периметр рівнобедреного трикутника дорівнює 28 см, а бічна сторона - 10 см*). Аналізатор просто пропускає значення 10 см з аналізу, а без цієї інформації вирішення задачі стає неможливим.

По-друге, навіть якщо замінити тире після слова «бічна сторона» на слово “дорівнює” PullEnti не може зв'язати чисто 10 разом з підметом «бічна сторона» (див. рисунок 3.2).

Приклад аналізу задачі:

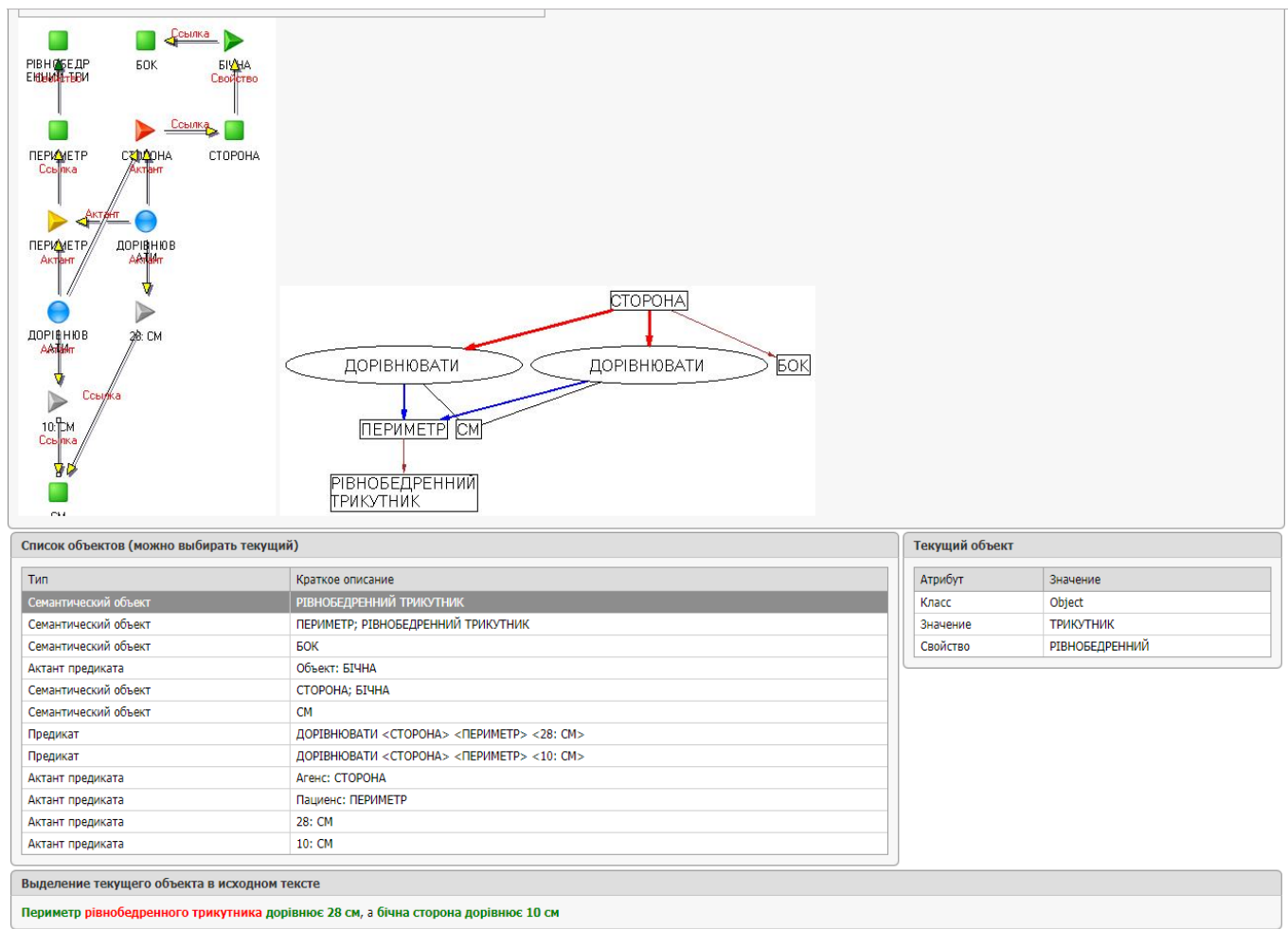


Рисунок 3.2 – Результат аналізу за допомогою бібліотеки PullEnti

Отже, використання даної бібліотеки для парсингу текстів геометричних задач не є можливим. І тут виникає проблема, для того щоб розпарсити текст задачі потрібен дійсно хороший семантичний аналізатор української мови, який би виділяв всі частини мови в задачі та пов'язував всі частини речень до купи. На жаль, такого аналізатора в відкритому доступі не було знайдено, а писати подібну бібліотеку з нуля потребує багато часу і зусиль. Тим паче, подібна робота є вже окремою курсовою роботою про аналіз природньої мови

в текстах. Саме тому подібні задачі досі вирішують живі люди, а не програмні застосунки.

З вищезазначених причин готова програма не здатна вирішувати будь-яку задачу з шкільного курсу геометрії.

Тому на даний момент програма здатна вирішити дві задачі одну з теми “трикутники” та одну (в різних варіаціях) з теми “паралельні прямі” за допомогою ручного парсингу умови задач.

Реалізація алгоритму

Спочатку текст задачі зчитується через графічний інтерфейс написаний на Java Swing (див. рисунок 3.1). Далі текст розбивається спочатку на окремі рядки, а далі на окремі лексеми та записується в стек (див. рисунок 3.3)

```
Stack<String> exercise = new Stack<>();  
for (String line :  
    task.split( regex: "\n")) {  
    String[] wordsArray = line.split( regex: " ");  
    for (String word :  
        wordsArray) {  
        exercise.push(word);  
    }  
}
```

Рисунок 3.3 – Розбивання тексту задачі на лексеми та запис в стек.

Після цього програма проходить по стеку, перевіряючи чи присутні лексеми, які відносяться до теми “трикутники” або “паралельні прями” (див. рисунок 3.4)


```

private static String analyze(Stack<String> exercise) {
    String answer = "";
    Shape sh = new Shape() {
        @Override
        public double perimeter() {
            return 0;
        }

        @Override
        public double area() {
            return 0;
        }
    };
    ParallelLines parallelLines = null;
    for (int i = 0; i < exercise.size(); ++i) {
        String fst = exercise.get(i);
        if (fst.equals("Один")) {
            i = i + 2;
            fst = exercise.get(i);
            if (fst.equals("односторонніх")) {
                parallelLines = new ParallelLines();
                while (!isNumeric(fst)) {
                    if (i < exercise.size() - 1)
                        fst = exercise.get(++i);
                    else return "";
                }
            }
            if (exercise.get(i - 1).equals("у") || exercise.get(i - 1).equals("в")) {
                if (exercise.get(i + 2).equals("більший")) {
                    parallelLines.setAngleC(180 / (Double.parseDouble(fst) + 1));
                } else
                if (exercise.get(i + 2).equals("менший")) {
                    parallelLines.setAngleE(180 / (Double.parseDouble(fst) + 1));
                }
            } else if (exercise.get(i - 1).equals("на")) {
                if (exercise.get(i + 2).equals("більший")) {
                    parallelLines.setAngleC((180 - Double.parseDouble(fst)) / 2);
                } else
                if (exercise.get(i + 2).equals("менший")) {
                    parallelLines.setAngleE((180 - Double.parseDouble(fst)) / 2);
                }
            }
        }
    }
}

```

Рисунок 3.4 – Аналіз лексем з стеку.

В процесі аналізу створюється об'єкт потрібного, для вирішення задач, класу та задаються значення, які відомі. Далі перевіряється, що

саме треба знайти після лексеми «Знайдіть» і викликається гетер класу, з яким ми на даний момент працюємо. (див. рисунок 3.5)

```
//looking for question
if (fst.equals("Знайдіть")) {
    int g = ++i;
    String snd = exercise.get(g);
    String trd = exercise.get(g + 1);
    if (snd.equals("основу")) {
        answer = "Основа =" + ((MyTriangle) sh).getSideC();
    }
    if (trd.equals("кути") || trd.equals("кути.")) {
        if (parallellines != null)
            answer = "Перший кут: " + parallellines.getAngleC() + ", Другий кут: " + parallellines.getAngleE();
    }
}
```

Рисунок 3.5 – Перевірка на те, що саме потребується знайти.

Знаючи чому дорівнює один з кутів, легко можна знайти всі інші кути, які прилягають до січної. При виклику гетера якогось кута з класу “ParallelLines” кут автоматично обраховується, якщо відомий будь-який інший кут (див. рисунок 3.6).

```
public double getAngleE() {
    if (angleA != 0)
        return angleA;
    if (angleB != 0)
        return 180 - angleB;
    if (angleC != 0)
        return 180 - angleC;
    if (angleD != 0)
        return angleD;
    if (angleE != 0)
        return angleE;
    if (angleF != 0)
        return 180 - angleF;
    if (angleG != 0)
        return 180 - angleG;
    if (angleH != 0)
        return angleH;
    return 0;
}
```

Рисунок 3.6 – Обрахунок кута через інший кут.

Після цього метод “analyze” повертає стрічкою отриману відповідь. У разі порожньої відповіді користувачу виводиться повідомлення про те, що програма ще не вміє вирішувати подібні задачі (див. рисунок 3.7).

```
analyzeButton.addActionListener(e -> {  
    String task = taskTextField.getText();  
    if(task.length() == 0){  
        JOptionPane.showMessageDialog( parentComponent: UI.this, (message: "Task field can't be empty", title: "Error", JOptionPane.ERROR_MESSAGE);  
        return;  
    }  
    else{  
        Analyzer analyzer = new Analyzer();  
        String answer = analyzer.start(task);  
        if(answer.length() == 0){  
            JOptionPane.showMessageDialog( parentComponent: UI.this,  
                (message: "Unfortunately, the program is not yet able to solve such tasks", title: "Error", JOptionPane.ERROR_MESSAGE);  
            return;  
        }  
        answerTextArea.setText(analyzer.start(task));  
    }  
});
```

Рисунок 3.7 – Вивід результату.

Огляд структури програми

Програма складається класів фігур, аналізатора і класу, що відповідає за графічний інтерфейс користувача.

Структура класів:

1. Shape – інтерфейс для всіх класів фігур, має два віртуальних методи `area()` та `perimeter()`.
2. Triangle – клас трикутника для вирішення задач з про трикутники. Містить всі поля, які відповідають всім властивостям, які може мати трикутник. Має багато конструкторів для різної кількості відомих аргументів. Також містить `enum Type` для вказівки типу трикутника або для знаходження типу трикутника по відомих
3. Dot – клас точки є основою для класу `LineSegment` (відрізок). Точка має свою назву, а також координати (якщо такі вказані в умові задачі).
4. `LineSegment` – клас відрізків, складається з двох точок і довжини. Основа для класу `Triangle`.
5. `Line` – клас прямої, має тільки назву прямої використовується класом `ParallelLines`.
6. `ParallelLines` – клас паралельних прямих для вирішення задач з теми геометрії «паралельні прямі». Складається з трьох прямих (дві паралельні та січна) та восьми кутів (всі кути, що утворюються при перетині двох прямих січною).
7. `Analyzer` – клас, що проводить аналіз введеної задачі, розбиває її на лексеми, створює екземпляри класів, які необхідні для вирішення задачі та викликає методи цих класів.

8. UI – клас, що наслідується від JFrame. Відповідає за відображення графічного вікна куди відбувається ввід задачі та вивід результату

Також є ще багато різних атрибутів та методів у всіх класах. Але описувати кожне поле і кожен метод є недоцільним.

Висновки

Отже, в роботі описані труднощі пов'язані з створенням готової системи задля вирішення математичних задач, поданих штучною мовою. Проаналізовано тривіальні принципи вирішення подібних задач, якими керуються учні при вирішенні подібних задач і як ці принципи можна застосувати в готовій програмі.

Проаналізовано тип мови в математичних задачах і які дії необхідно зробити для аналізу умови цих задач. Розглянуто основні відмінності між природними та штучними мовами. Наведено нормативні принципи штучних мов, яким підпорядковуються всі штучні мови.

Також було розглянуто теоретичну частину, яка стосується онтологій і було з'ясовано, для яких цілей зазвичай створюються онтології. Розглянуто створену онтологію планіметричних фігур і які висновки можна отримати, спираючись на створену онтологію.

Побудовано алгоритм для розв'язання задач з геометрії для програмного застосунку, а також описані властивості, які мають бути в гарному алгоритмі.

Розглянуто бібліотеку PullEnti та наведено причини, з яких використання даної бібліотеки для розробки програмного застосунку не є можливим.

На завершення розглянуто структуру класів програми для вирішення задач з геометрії. Сама програма є своєрідною “заготовкою” без семантичного аналізатора української мови. І її легко можна перетворити в повністю готовий програмний застосунок, якщо розробити такий аналізатор.

Список літератури

1. Онлайн калькулятор периметру квадрату.
http://ua.onlinemschool.com/math/assistance/figures_perimeter/square/.
2. Онлайн калькулятор площі трикутника дев'ятьма способами.
http://ua.onlinemschool.com/math/assistance/figures_area/triangle/.
3. Мерзляк А. Г. «Геометрія» підручник для 7 класу загальноосвітніх навчальних закладів / Полонський В. Б., Якір М. С. – Х. : Гімназія, 2015. – 224 с. : іл.
4. Природні та штучні мови. Stud.com.ua
https://stud.com.ua/57108/logika/prirodni_shtuchni_movi.
5. Добров Б. В., Иванов В.В., Лукашевич Н.В., Соловьев В.Д.
Онтологии и тезаурусы: модели, инструменты, приложения. — М.: Бином. Лаборатория знаний, 2009. — 173 с. — ISBN 978-5-9963-0007-5.
6. Thomas R. Gruber / A Translation Approach to Portable Ontology Specifications - Appeared in Knowledge Acquisition, 5(2):199-220, 1993.
7. DEBORAH L. McGUINNESS, JON R. WRIGHT / Conceptual modelling for configuration: A description logic-based approach — Published online by Cambridge University Press: 01 September 1998, Volume 12, Issue 4, pp. 333-344.
8. Gregg R. Yost, Thomas R. Rothenfluh / Configuring elevator systems — International Journal of Human-Computer Studies, Volume 44, Issues 3–4, March 1996, Pages 521-568.
9. Deborah L. McGuinness, Richard Fikes, James Rice, Steve Wilder / The Chimaera Ontology Environment — In the Proceedings of the The Seventeenth National Conference on Artificial Intelligence (AAAI 2000), Austin, Texas, July 30 - August 3, 2000.

10. Поняття алгоритму. Властивості алгоритмів

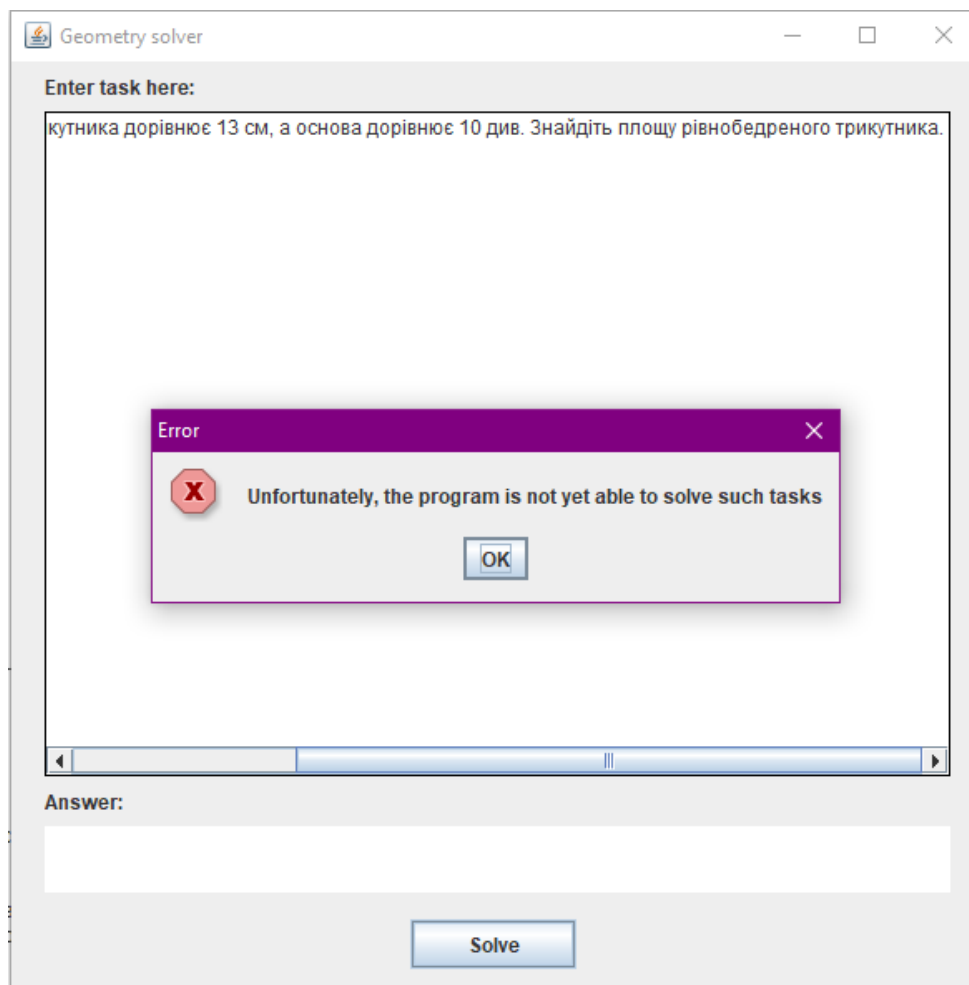
<https://sites.google.com/site/vchimoinformatikurazom/zavdanna/pontata-algoritmu>.

11. Бібліотека PullEnti

<http://www.pullenti.ru/Default.aspx>

Додаток А
(довідниковий)

Повідомлення про не вміння програми вирішувати введену задачу



Додаток Б
(довідниковий)
Вирішена задача на тему “трикутники”

Geometry solver

Enter task here:

Рівнобедреного трикутника дорівнює 28 см, а бічна сторона - 10 см. Знайдіть основу трикутника.

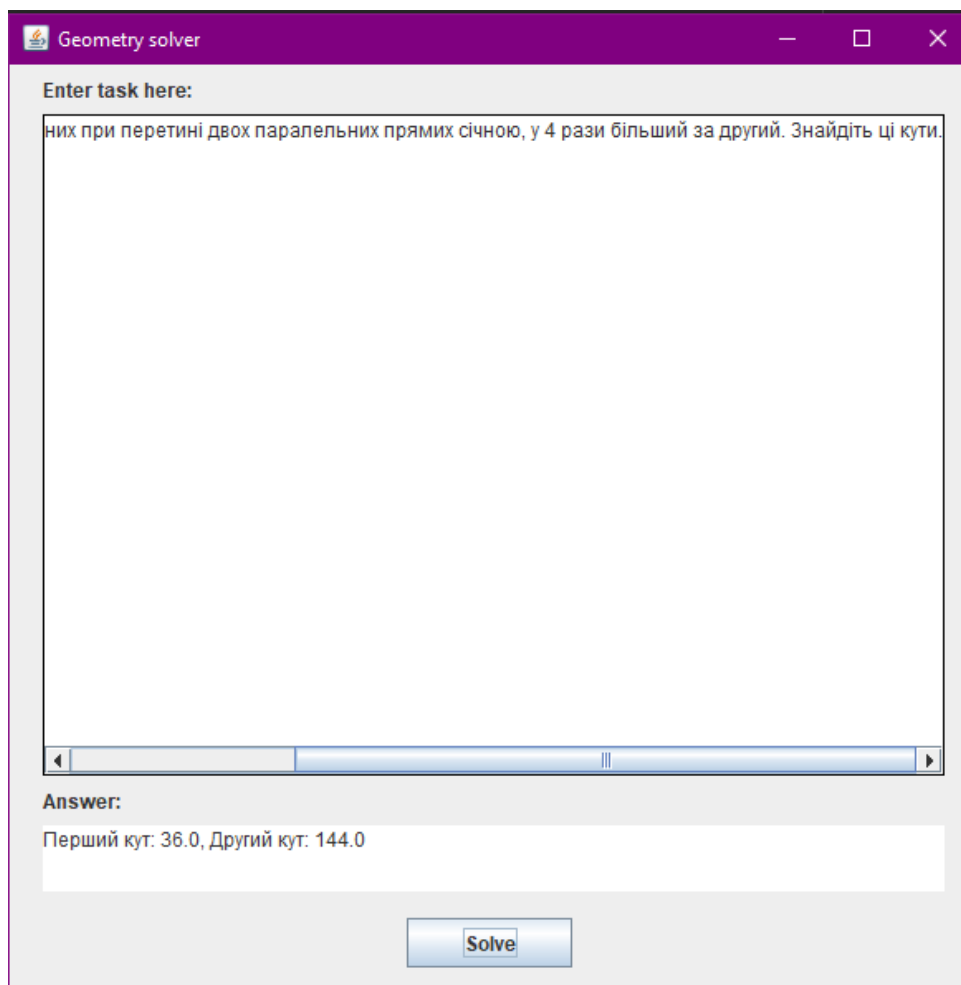
Answer:

Основа =8.0

Solve

Додаток В
(довідниковий)

**Вирішена задача на тему “паралельні прямі” з
формулюванням «в 4 рази більший»**



The screenshot shows a window titled "Geometry solver" with a purple header bar. Below the header, there is a text input area labeled "Enter task here:". The task text, entered in the input area, is: "них при перетині двох паралельних прямих січною, у 4 рази більший за другий. Знайдіть ці кути." Below the input area is a horizontal scrollbar. Under the scrollbar, the word "Answer:" is displayed. Below "Answer:", the solution is provided: "Перший кут: 36.0, Другий кут: 144.0". At the bottom center of the window is a button labeled "Solve".

Geometry solver

Enter task here:

них при перетині двох паралельних прямих січною, у 4 рази більший за другий. Знайдіть ці кути.

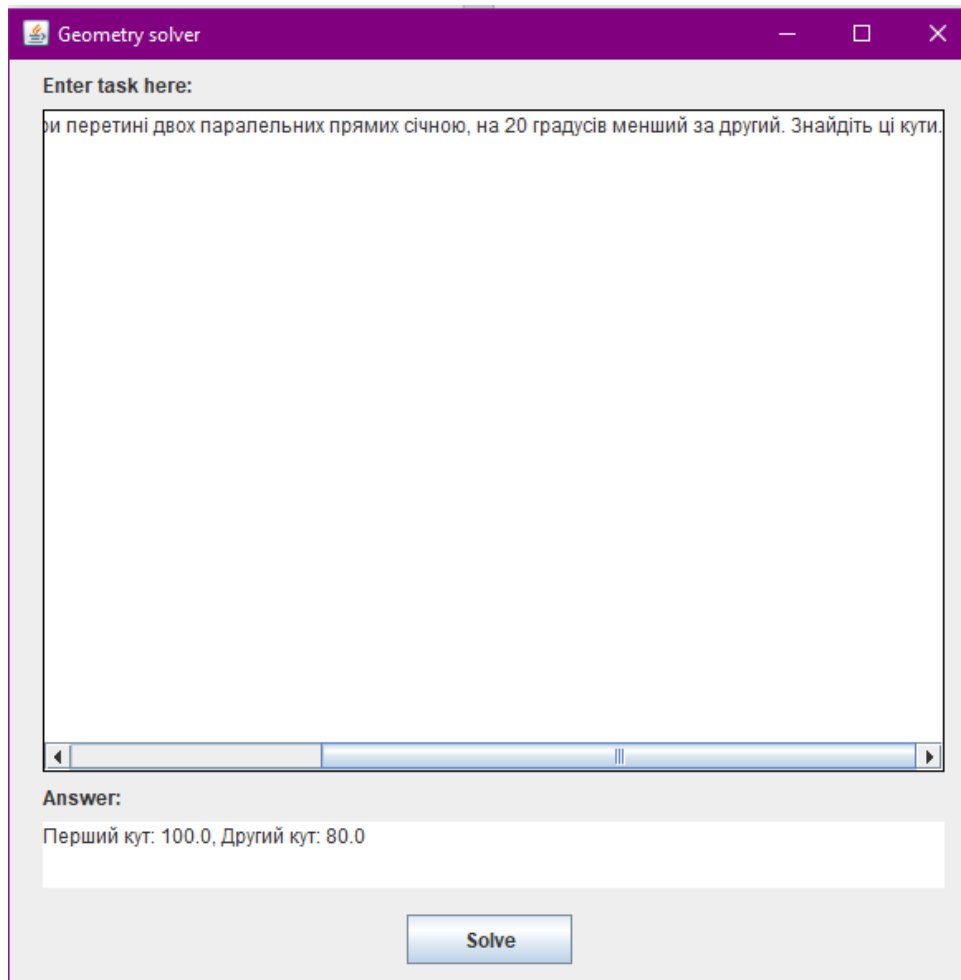
Answer:

Перший кут: 36.0, Другий кут: 144.0

Solve

Додаток Г
(довідниковий)

**Вирішена задача на тему “паралельні прямі” з
формулюванням «на 20 градусів менший»**



The screenshot shows a window titled "Geometry solver" with a purple header bar. Inside the window, there is a text input area with the prompt "Enter task here:". The input area contains the text: "При перетині двох паралельних прямих січною, на 20 градусів менший за другий. Знайдіть ці кути." Below the input area is a horizontal scrollbar. Underneath the scrollbar, the text "Answer:" is displayed, followed by a white box containing the solution: "Перший кут: 100.0, Другий кут: 80.0". At the bottom center of the window is a blue button labeled "Solve".

Geometry solver

Enter task here:

При перетині двох паралельних прямих січною, на 20 градусів менший за другий. Знайдіть ці кути.

Answer:

Перший кут: 100.0, Другий кут: 80.0

Solve

Додаток Г
(довідниковий)

Натиснута кнопка з порожнім полем для вводу задачі

