

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка
до дипломного проєкту (роботи)

магістр
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
МЕТОДІВ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ.
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS
OF TEXT DATA ANALYSIS

Виконав: студент 2 курсу, групи КНТ-119м
спеціальності

121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Дедик А. П.

(прізвище та ініціали)

Керівник Олійник А.О., к.т.н., доцент

(прізвище та ініціали)

Рецензент Гофман Є.О., к.т.н.

(прізвище та ініціали)

2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет _____ ІРЕ, КНТ _____
 Кафедра _____ програмних засобів _____
 Ступінь вищої освіти _____ магістр _____
 Спеціальність _____ 121 Інженерія програмного забезпечення _____
 (код і назва)
 Освітня програма (спеціалізація) _____ Інженерія програмного забезпечення _____
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри проф., д.т.н.

С. О. Субботін

“ ” _____ 2020 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

_____ Дєдика Антона Павловича _____

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) _____ Дослідження та програмна реалізація методів
аналізу текстових даних. Research and Software Implementation of Methods of
Text Data Analysis _____

керівник проєкту (роботи) _____ Олійник А.О., к.т.н., доцент _____,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “03” листопада 2020 року
№301

2. Строк подання студентом проєкту (роботи) _____ 5 грудня 2020 року _____

3. Вихідні дані до проєкту (роботи) _____ технічне _____ завдання _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _____ 1. Аналіз предметної області. 2. Вибір та обґрунтування
структури системи, яка проєктується. 3. Основні рішення щодо реалізації
компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора.
6. Економіко-організаційна частина. 7. Охорона праці та безпека у
надзвичайних ситуаціях. _____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
 _____ Слайди презентації _____

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1–5	Олійник А.О., доцент		
6	Пожуєва Т.О., професор		
7	Коробко О.В., ст. викладач		
Нормо-контроль	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання 02.10.20

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	1 тиждень	
2	Розробка та затвердження технічного завдання	2 тиждень	Технічне завдання
3	Проектування структури системи	3-4 тиждень	Структурна схема
4	Розробка схеми функціонування системи та модулів програми	5-6 тиждень	Функціональна схема. Модулі
5	Створення програмного коду системи	7-8 тиждень	Текст програми
6	Тестування та відлагодження програми	9 тиждень	Працездатна система
7	Розробка розділів пояснювальної записки	10 тиждень	ПЗ
8	Розробка слайдів презентації	11-12 тиждень	Слайди презентації
9	Захист роботи	13 тиждень	

Студент(ка)

(підпис)

Дедик А.П.

(прізвище, ініціали)

Керівник проєкту (роботи)

(підпис)

Олійник А.О.

(прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
126 с., 27 рис., 4 додатки, 26 джерел.

АНАЛІЗ ДАНИХ, ДЕРЕВО РОЗВ'ЯЗКІВ, ПРОГРАМНЕ
ЗАБЕЗПЕЧЕННЯ, СЕМАНТИЧНИЙ АНАЛІЗ, СИНТАКСИЧНИЙ АНАЛІЗ,
ТЕКСТ.

Об'єкт дослідження – процес обробки текстових даних.

Предмет дослідження – методи аналізу текстових даних.

Мета роботи – дослідження та розробка засобів аналізу текстових даних.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, середовище розробки Matlab, персональний комп'ютер з монітором, клавіатурою та маніпулятором типу «миша», що функціонує під управлінням операційної системи.

Результати. Створено програмні засоби для аналізу текстових даних. Розроблена програмна система може успішно експлуатуватися на персональних комп'ютерах під управлінням операційних систем та використовуватися для обробки та аналізу текстової інформації.

Висновки. Розроблено програмне забезпечення для аналізу текстових даних. Проаналізовано предметну область, розроблено технічне завдання для реалізації програми, обрано середовище розробки, розроблено структурну схему програми, виконано конструювання програмного забезпечення аналізу текстових даних, здійснено тестування розробленого програмного забезпечення.

Галузь використання – програмні засоби аналізу текстових даних.

ABSTRACT

Explanatory note to the final qualifying work of the magister: 126 pages, 27 figures, 4 appendixes, 26 sources.

DATA MINING, DECISION TREE, SOFTWARE, SEMANTIC ANALYSIS, SYNTAX ANALYSIS, TEXT.

Object of study is a process of text data processing.

Subject of study are methods of text data analysis.

The purpose of the work is to study and develop a text data analysis software tools.

Materials, methods, and tools: structured and object-oriented programming, programming language Matlab, computer with a monitor, keyboard and mouse manipulator operating under the control of the operating system.

Results. The software in the form of a computer game based on intelligent technologies has been created. The developed software system can successfully operate on personal computers running operating system and can be used for processing and analysis of textual information.

Conclusions. Software for text data analysis has been developed. The subject area is analyzed, the technical task for program realization is developed, the development environment is chosen, the structural scheme of the program is developed, the software for text data analysis is executed, the developed software is tested.

The field of use are software tools for text data analysis.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	10
1.1 Дослідження формальних граматик.....	10
1.2 Синтаксичний аналіз.....	12
1.3 Розширені граматики	15
1.4 Семантична інтерпретація.....	23
1.5 Неоднозначність та усунення неоднозначності	29
1.6 Висновки за розділом 1	33
2 Вибір та обґрунтування структури системи, яка проєктується.....	34
2.1 Розробка методу синтаксичного аналізу на основі діаграмного підходу .	34
2.2 Вибір інструментарію розробки програмного забезпечення	39
2.3 Структурна схема програми.....	40
2.4 Опис модулів системи, що розробляється.....	41
2.5 Висновки за розділом 2	43
3 Основні рішення щодо реалізації компонентів системи.....	45
3.1 Функціональна схема програми	45
3.2 Експерименти і результати.....	47
3.3 Висновки за розділом 3	49
4 Керівництво програміста	50
4.1 Призначення й умови застосування програми.....	50
4.2 Характеристики програми.....	50
4.3 Звернення до програми.....	51
4.4 Вхідні і вихідні дані	51
4.5 Повідомлення.....	51
5 Керівництво оператора	53
5.1 Призначення програми	53

5.2 Умови виконання програми	53
5.3 Виконання програми	53
5.4 Приклад роботи з програмою	54
5.5 Повідомлення оператору	58
6 Економіко-організаційна частина.....	60
6.1 Планування розробки програмного продукту.....	60
6.2 Визначення витрат на розробку програми	61
6.3 Розрахунок економічної ефективності програмного продукту.....	68
7 Охорона праці та безпека у надзвичайних ситуаціях.....	71
7.1 Аналіз потенційних небезпек.....	71
7.2 Заходи щодо забезпечення безпеки.....	72
7.3 Заходи щодо виробничої санітарії та гігієни праці	74
7.4 Заходи безпеки у надзвичайних ситуаціях	78
7.4.1 Заходи з пожежної безпеки	78
7.4.2 Заходи з цивільного захисту	79
Висновки	83
Перелік джерел посилання	85
Додаток А Технічне завдання	88
Додаток Б Опис програми	92
Додаток В Текст програми	95
Додаток Г Слайди презентації	119

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ОС – операційна система;

ПЗ – програмне забезпечення;

ПО – предметна область;

ПК – персональний комп'ютер;

ПП – програмний продукт;

ШІ – штучний інтелект.

ВСТУП

У сучасному світі з усебічним поширенням інформаційних технологій актуальним стає створення програмного забезпечення, здатного спілкуватися з користувачем та розуміти його природну мову, зокрема аналізувати інформацію, подану користувачем у текстовому вигляді.

Обробка природної мови – загальний напрямок штучного інтелекту і математичної лінгвістики, який вивчає проблеми комп'ютерного аналізу і синтезу природних мов. Стосовно штучного інтелекту аналіз означає розуміння мови, а синтез – генерацію грамотного тексту. Вирішення цих проблем буде означати створення більш зручної форми взаємодії комп'ютера і людини. Розуміння природної мови іноді вважають завданням штучного інтелекту, оскільки розпізнавання живої мови вимагає величезних знань системи про навколишній світ і можливості з ним взаємодіяти. Розгляду цієї галузі штучного інтелекту і присвячена ця робота.

У роботі вирішується актуальне завдання дослідження та програмної реалізації методів аналізу текстових даних. Об'єктом дослідження є процес обробки текстових даних. Мета виконання роботи полягає в дослідженні та розробці засобів аналізу текстових даних.

Для досягнення поставленої мети необхідно було розв'язати такі завдання:

- дослідження методів аналізу текстових даних;
- розробка технічного завдання для реалізації програми;
- вибір середовища розробки програмного забезпечення;
- розробка методу синтаксичного аналізу на основі діаграмного підходу;
- розробка структурної схеми програми;
- конструювання програмного забезпечення для аналізу текстових даних;
- тестування розробленого програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження формальних граматики

Формальною граматикою називають спосіб опису формальної мови, тобто виділення деякої підмножини з множини всіх можливих слів деякого кінцевого алфавіту.

Визначимо формальну граматику для невеликої підмножини англійської мови (англійську мову обрано, оскільки вона є однією з ієроархивних рекурсивними у сучасному світі), яка придатна для складання тверджень про світ. Ми будемо називати цю мову ϵ_0 . У наступних розділах ϵ_0 буде вдосконалена з тією метою, щоб вона стала трохи ближче до справжньої англійської мови [1].

Спочатку визначимо словник, або список допустимих слів. Ці слова групуються за категоріями або частинами мови, які знайомі всім користувачам тлумачних словників: іменники, займенники та імена власні позначають предмети, дієслова позначають події, прикметники модифікують іменники, а прислівники модифікують дієслова. Категоріями, які, можливо, менш знайомі деяким читачам, є артиклі (такі як «the»), прийменники (наприклад, «in») та сполучники (на зразок «and»). Приклад невеликого словника мови ϵ_0 наведено на рис. 1.1.

Noun → stench | breeze | glitter | nothing | agent
 | wumpus | pit | pits | gold | east | ...
 Verb → is | see | smell | shoot | feel | stinks | go
 | grab | carry | kill | turn | ...
 Adjective → right | left | east | dead | back | smelly | ...
 Adverb → here | there | nearby | ahead | right | left
 | east | south | back | ...
 Pronoun → me | you | I | it | ...
 Name → John | Mary | Boston | Aristotle | ...
 Article → the | a | an | ...
 Preposition → to | in | on | near | ...
 Conjunction → and | or | but | ...
 Digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ...

Рисунок 1.1 – Словник мови ϵ_0

Майже кожна з категорій закінчується трьома крапками (...) для вказівки на те, що в цій категорії є й інші слова. Що стосується іменників, дієслів, прикметників і прислівників, то перерахувати їх повністю неможливо в принципі, оскільки існують десятки тисяч елементів кожного класу, крім того постійно додаються все нові і нові, які виникають в процесі використання та модифікації мови її носіями. Ці чотири категорії називаються відкритими класами. Інші категорії (займенники, артиклі, прийменники і союзи) називаються закритими класами. Вони включають невелику кількість слів (від декількох слів до декількох десятків слів), які можна в принципі перерахувати повністю. Склад закритих класів змінюється протягом століть. Наприклад, займенники «thee» і «thou» широко використовувалися в англійській мові в XVII столітті, потім в XIX столітті їх частота пішла на спад, а в наші дні вони, як правило, не використовуються [2].

На наступному етапі необхідно забезпечити об'єднання слів в словосполучення. Будемо використовувати п'ять іво рекурсивних символів для визначення словосполучень різних типів: речення (Sentence – S), іменне словосполучення (Noun Phrase – NP), дієслівне словосполучення (Verb Phrase – VP), прийменникове словосполучення (Prepositional Phrase – PP) і відносне речення (Relative Clause – RelClause). Граматика мови ϵ_0 приведена на рис. 1.2, де для кожного правила підстановки наведений приклад [3].

Проте ця граматика не тільки генерує правильні речення, а й допускає перевиробництво, тобто пропонує речення, які є не грамотними, такі як «Me go Boston» і «I smell pit gold wumpus nothing east». Крім того, ця граматика допускає недовиробництво – вона відкидає багато правильних англійських речень, такі як «I think the wumpus is smelly».

S → NP VP	I + feel a breeze
S Conjunction S	I feel a breeze + and +
	I smell a wumpus
NP → Pronoun	I
Name	John
Noun	pits
Article Noun	the + wumpus
Digit Digit	3 4
NP NP	the wumpus + to the east
NP RelClause	the wumpus + that is smelly
VP → Verb	stinks
VP NP	feel + a breeze
VP Adjective	is + smelly
VP PP	turn + to the east
VP Adverb	go + ahead
PP → Preposition NP	to + the east
RelClause → that VP	that + is smelly

Рисунок 1.2 – Граматика мови ϵ_0 з прикладами словосполучень, що ілюструють кожне правило

1.2 Синтаксичний аналіз

Синтаксичний аналіз – процес пошуку синтаксичного дерева для даного конкретного вхідного рядка. Це означає, що виклик функції синтаксичного аналізу Parse, такий як $\text{Parse}(\text{“the wumpus is dead”}, \epsilon_0, S)$ повинен привести до отримання дерева синтаксичного аналізу з коренем S, листям якого є слова «the wumpus is dead», а внутрішніми вузлами – ієрархічними символами граматики ϵ_0 . У вигляді лінійного тексту це дерево може бути записане таким чином: $[S: [NP: [Article: the][Noun: wumpus]] [VP: [Verb: is][Adjective: dead]]]$.

Синтаксичний аналіз може розглядатися як процес пошуку дерева синтаксичного аналізу. Існують два протилежні способи завдання відповідного простору пошуку (а також множини проміжних варіантів таких способів). По-перше, можна почати з символу S і шукати дерево, листям

якого є відповідні слова. Такий спосіб називається низхідним синтаксичним аналізом (оскільки символ S зображується у верхній частині малюнка, на якому показано дерево, повернене коренем вгору). По-друге, можна почати зі слів і виконувати пошук дерева з коренем S . Такий метод називається висхідним синтаксичним аналізом. Метод рекурсивного спуску може бути точно визначено як завдання пошуку відповідно до приведенного нижче опису.

Крок 1. Початковий стан являє собою дерево синтаксичного аналізу, що складається з кореня S і невідомих дочірніх вузлів: $[S: ?]$. Взагалі кажучи, кожний стан в просторі пошуку також є деревом синтаксичного аналізу.

Крок 2. Функція визначення наступника вибирає в дереві івно рекурсивний вузол з невідомими дочірніми вузлами. Потім в граматичі здійснюється пошук правил, які мають кореневу мітку вузла, що знаходиться в лівій частині. Для кожного такого правила створюється стан-наступник, в якому символ $?$ замінюється списком, що відповідає правій частині правила. Наприклад, в граматичі ϵ_0 є два правила для S , тому дерево $[S: ?]$ має бути замінене наступними двома наступниками: $[S: [S: ?][Conjunction: ?][S: ?]]$ $[S: [NP: ?][VP: ?]]$

Другий з цих наступників має сім наступників, по одному для кожного правила підстановки NP . У перевірці мети перевіряється, які листя дерева синтаксичного аналізу точно відповідають вхідному рядку, без невідомих і неохоплених вхідних даних [4].

Одна суттєва проблема при низхідному синтаксичному аналізі виникає, коли доводиться стикатися з так званими івно рекурсивними правилами, тобто правилами в формі $X \rightarrow X \dots$. При пошуку в глибину застосування такого правила може призвести до того, що заміна X на $[X: x \dots]$ буде здійснюватися в нескінченному циклі. А при пошуку завширшки можна буде успішно знайти варіанти синтаксичного аналізу для допустимих речень, але при наявності неприпустимого речення може виникнути така ситуація, що програма не зможе вийти з нескінченного простору пошуку.

Нижче наведено опис висхідного синтаксичного аналізу як завдання пошуку.

Крок 1. Початковим станом є список слів у вхідному рядку, де кожне слово розглядається як дерево синтаксичного аналізу тільки з одним листовим вузлом, наприклад [the , wumpus , is , dead]. Взагалі кажучи, кожний стан в просторі пошуку являє собою список дерев синтаксичного аналізу.

Крок 2. За допомогою функції визначення наступника виконується пошук в кожній позиції, і списку дерев, і в кожній правій частині правила граматики. Якщо підпоследовність списку дерев, що починається з і, узгоджується з правою частиною, то ця підпоследовність замінюється новим деревом, категорією якого є ліва частина правила, а дочірніми вузлами – ця підпоследовність. Під «узгодженням» мається на увазі, що категорія вузла є такою ж, як і елемент в правій частині. Наприклад, правило Article → the узгоджується з підпоследовністю, яка складається з першого вузла в списку [the , wumpus , is , dead], тому станом-наступником стає [[Article : the] , wumpus , is , dead].

Крок 3. У перевірці мети перевіряється наявність стану, що представляє собою єдине дерево з коренем S.

Приклад висхідного синтаксичного аналізу наведено у таблиці 1.1.

Таблиця 1.1 – Трасування висхідного синтаксичного аналізу рядка «The wumpus is dead»

Етап	Список узлов	Підпоследовність	Правило
Init	the wumpus is dead	the	Article → the
2	Article wumpus is dead	wumpus	Noun → wumpus
3	Article Noun is dead	Article Noun	NP → Article Noun
4	NP is dead	is	Verb → is
5	NP Verb dead	dead	Adjective → dead

Продовження таблиці 1.1

Етап	Список узлов	Підпоследовність	Правило
6	NP Verb Adjective	Verb	$VP \rightarrow Verb$
7	NP VP	VP Adjective	$VP \rightarrow VP$ Adjective
8	S	NP VP	$S \rightarrow NP VP$
Goal	S		

І спадний, і висхідний синтаксичний аналіз може виявитися неефективним через те, що окремі етапи синтаксичного аналізу різних поєднань можуть комбінуватися різними способами. І в тій, і в іншій процедурі можуть виникати непродуктивні витрати часу, пов'язані з пошуком в тих частинах простору станів, які не дозволяють отримати необхідний результат. При низхідному синтаксичному аналізі іноді створюються проміжні вузли, яким так і не вдається зв'язатися зі словами, а при висхідному синтаксичному аналізі створюються часткові фрагменти синтаксичного аналізу слів, які неможливо перетворити в кореневий вузол S.

Але навіть якщо б існувала ідеальна евристична функція, що дозволяє здійснювати пошук без непотрібних відступів, ці алгоритми все одно були б неефективними, оскільки для деяких пропозицій кількість дерев синтаксичного аналізу вимірюється експоненційною залежністю [5].

1.3 Розширені граматики

Як було показано у підрозділі 1.2, проста граMATика для мови ϵ_0 генерує речення «I smell a stench» (Я відчуваю неприємний запах) і багато інших речень англійською мовою. На жаль, вона генерує також багато словосполучень, які не є реченнями, таких як «Me smell a stench». Для запобігання цієї проблеми в цій граматиці потрібно було б врахувати, що

слово «me» не є допустимим словосполученням NP при його використанні як підмет деякого речення. Як кажуть лінгвісти, займенник «I» знаходиться в називному відмінку, а «me» – в об’єктному відмінку. Після того як врахована наявність відмінка, стає очевидно, що граMATика ϵ_0 не є контекстно-вільною, оскільки не можна стверджувати, що будь-яке словосполучення NP еквівалентно будь-якому іншому незалежно від контексту. Цей недолік можна виправити, ввівши нові категорії, такі як NP_S і NP_0 , для позначення іменникових словосполучень в називному і об’єктному відмінку відповідно. Для цього також потрібно розділити категорію займенників Pronoun на дві категорії – $Pronoun_S$ (яка включає «I») і $Pronoun_0$ (яка включає «me»). У верхній частині рис. 1.3 показана повна граMATика Bunny, підготовлена з урахуванням угоди про відмінку. Ми будемо називати отриману в результаті мову ϵ_1 . Зверніть увагу на те, що довелося продублювати всі правила NP, ввівши одне правило для NP_S , а інше – для NP_0 .

$$\begin{aligned}
 S &\rightarrow NPS VP \mid \dots \\
 NP_S &\rightarrow Pronoun_S \mid Name \mid Noun \mid \dots \\
 NP_0 &\rightarrow Pronoun_0 \mid Name \mid Noun \mid \dots \\
 VP &\rightarrow VP NP_0 \mid \dots \\
 PP &\rightarrow Preposition NP_0 \\
 Pronoun_S &\rightarrow I \mid you \mid he \mid she \mid it \mid \dots \\
 Pronoun_0 &\rightarrow me \mid you \mid him \mid her \mid it \mid \dots
 \end{aligned}$$

$$\begin{aligned}
 S &\rightarrow NP \text{ (Subjective)} VP \mid \dots \\
 NP \text{ (case)} &\rightarrow Pronoun \text{ (case)} \mid Name \mid Noun \mid \dots \\
 VP &\rightarrow VP NP \text{ (Objective)} \mid \dots \\
 PP &\rightarrow Preposition NP \text{ (Objective)} \\
 Pronoun \text{ (Subjective)} &\rightarrow I \mid you \mid he \mid she \mid it \mid \dots \\
 Pronoun \text{ (Objective)} &\rightarrow me \mid you \mid him \mid her \mid it \mid \dots
 \end{aligned}$$

Рисунок 1.3 – Варіанти представлення граMATики

На жаль, мова ϵ_1 все ще допускає надмірність. В англійському та багатьох інших мовах потрібне узгодження між підметом і основним дієсловом речення. Наприклад, якщо підметом є «I», то речення «I smell» є

граматично правильним, а «I smells» – ні. Якщо ж підметом є «it», то вірно протилежне. В англійській мові відмінності, обумовлені узгодженням, є мінімальними: більшість дієслів мають одну форму для підмета, що представляє собою займенник третьої особи в однині («he», «she» або «it»), і другу форму для всіх інших комбінацій особи і числа. Є тільки один виняток: словосполучення «I am / you are / he is» мають три форми. Помноживши ці три різні форми на дві різні форми NP_S і NP_O , отримаємо шість форм NP . Якби кількість відмінностей була ще більше, то в кінцевому результаті загальна кількість варіантів виражалася б експоненційною залежністю.

Альтернативний підхід полягає в тому, що необхідно розширити існуючі правила граматики, а не вводити нові правила. Спочатку наведемо приклад того, як могли б виглядати розширені правила (див. нижню частину рис 1.3), а потім формально визначимо спосіб інтерпретації цих правил. Розширені правила допускають застосування параметрів в ϵ термінальних категоріях. На рис. 1.6 показано, як описати граматику ϵ_1 з використанням розширених правил. Категорії NP і $Pronoun$ мають параметр, що позначає їх відмінок (іменники не мають відмінка в англійській мові, але мають його у багатьох інших мовах). У правилі для S словосполучення NP має перебувати в називному відмінку, тоді як в правилах для VP і ϵ словосполучення NP має перебувати в об'єктному відмінку. Правило для NP приймає як параметр деяку змінну, $case$. Призначення цієї конструкції полягає в тому, що словосполучення NP може мати будь-який відмінок, але якщо замість цього словосполучення NP підставляється займенник $Pronoun$, то останнє повинно мати такий же відмінок. Застосування змінних, що дозволяє виключити необхідність прийняття рішення, коли відмінність, яке визначається змінною, не має значення, сприяє тому, що не доводиться збільшувати розмір безлічі правил в експоненційній залежності від кількості розглянутих характеристик.

Використовувана для такого доповнення формальна система називається граматиною певних виразів, або скорочено DCG (Definite Clause Grammar), оскільки кожне правило цієї граматики можна інтерпретувати, як

певний вираз в хорновській логіці. Спочатку ми покажемо, як можна інтерпретувати в якості певного вираження звичайні, нерозширені правила. Ми будемо розглядати кожен символ з позначенням категорії як предикат, заданий на рядках, тому вираз $NP(s)$ приймає істинне значення, якщо рядок s утворює деяке словосполучення NP . Наступне правило CFG: “ $S \rightarrow NP VP$ ” є скороченим позначенням для наступного певного виразу: “ $NP(s_1) \wedge VP(s_2) \Rightarrow S(s_1+s_2)$ ”.

Тут s_1+s_2 позначає конкатенацію двох рядків, тому дане правило говорить про те, що якщо рядок s_1 являє собою словосполучення NP , а рядок s_2 – словосполучення VP , то їх конкатенація є словосполучення S . Це означає, що дана інтерпретація точно відповідає тому, як ми вже інтерпретували це правило CFG. Важливо відзначити, що граматика певних виразів дозволяє розглядати синтаксичний аналіз як логічний висновок. Такий підхід відкриває можливість міркувати про мови і рядки за допомогою багатьох різних способів. Наприклад, він означає, що висхідний синтаксичний аналіз можна здійснювати за допомогою прямого логічного висновку, а метод рекурсивного спуску – за допомогою зворотнього логічного висновку. Як буде показано нижче, це також означає, що одна і та ж граматика може використовуватися і для синтаксичного аналізу, і для виробництва нових речень.

Але найважливішою перевагою підходу на основі DCG є те, що він дає можливість доповнювати символи категорій допоміжними параметрами, відмінними від звичайних строкових параметрів.

Але за таку зручність доводиться платити: в розпорядження укладача граматики передається вся потужність системи автоматичного доведення теорем, тому доводиться відмовлятися від гарантованого виконання синтаксичного аналізу за час $O(n^3)$. Задача синтаксичного аналізу граматики з розширеннями може виявитися NP-важкою або навіть нерозв’язною, в залежності від цих розширень.

Для того щоб забезпечити застосування граматики DCG, необхідно скористатися ще кількома прийомами, наприклад, потрібно знайти спосіб завдання термінальних символів, крім того, було б зручно мати можливість не вводити автоматичний строковий параметр. Врахувавши всі ці побажання, автори вирішили, що потрібно задати граматику певних виразів як описано нижче:

– запис $X \rightarrow Y Z \dots$ повинен бути перетворений в запис $Y(s_1)^{\wedge}Z(s_2)^{\wedge}.. \Rightarrow X(s_1+s_2+\dots)$;

– запис $X \rightarrow Y | Z | \dots$ повинен бути перетворений в запис $Y(s) \vee Z(s) \vee .. \Rightarrow X(s)$;

– в тому та у наведеному вище правилі будь-який ϵ термінальний символ Y може бути доповнений одним або декількома параметрами. Кожен параметр може являти собою змінну, константу або функцію від деяких параметрів. Після перекладу в форму певного вираження ці параметри повинні передувати строковому параметру;

– запис $\{P (\dots)\}$ може з'являтися в правій частині правила і буквально переводиться в $P(\dots)$. Це дозволяє укладачеві граматики вставляти перевірку для терміна $P(\dots)$ без необхідності додавання автоматичного строкового параметра;

– запис $X \rightarrow \text{word}$ перекладається як $x([\text{word}])$.

ГраMATика ϵ_1 є більш досконалою, але все ще допускає перевиробництво. Один з її недоліків полягає в тому, який спосіб застосовується для з'єднання дієслівних словосполучень з іншими словосполученнями. Потрібно, щоб вважалися прийнятними дієслівними словосполученнями тільки такі групи слів, як «give me the gold» (віддай мені золото) і «go to [1,2]» (вирушай в квадрат [1,2]). Всі ці словосполучення входять до складу мови ϵ_1 , але, на жаль, утворюються також словосполучення «go me the gold» і «give to [1,2]». Мова ϵ_1 дозволяє усунути подібні словосполучення VP шляхом явної вказівки на те, які словосполучення можуть йти за тими чи іншими дієсловами. Список з такою

інформацією називається списком субкатегоризації для дієслова. Ідея цього підходу полягає в тому, що категорія Verb підрозділяється на субкатегорії – на ту, в якій дієслова не мають об’єкта, в якій дієслова можуть мати один об’єкт, і т.п.

Для реалізації цієї ідеї для кожного дієслова передбачається список субкатегоризації, в якому перераховані доповнення дієслова. Доповненням називається обов’язкове словосполучення, яке йде за дієсловом в дієслівному словосполученні. Тому у словосполученні «Give the gold to me» словосполучення NP «the gold» і pp «to me» є доповненнями дієслова «give». Для запису такого списку застосовується наступна форма: “Verb([NP, PP]) → give | hand| ...”

Допустимо, що дієслово має декілька різних субкатегоризацій, так само, як і слово може належати до кількох різних категорій. Насправді дієслово «give» має список субкатегоризацій [NP, NP], як в словосполученні «Give me the gold». Таку ситуацію можна розглядати як ще один різновид неоднозначності, що вимагає усунення. У таблиці 1.2 наведені деякі приклади дієслів та їх списків субкатегоризації (або скорочено субкатегоризацій).

Таблиця 1.2 – Приклади дієслів з їх списками субкатегоризації

Дієслово	Список субкатегоризації	Приклад дієслівного стану
give	[NP, PP]	give the gold in 3 3 to me
	[NP, NP]	give me the gold
smell	[NP]	smell a wumpus
	[Adjective]	smell awful
	[PP]	smell like wumpus
is	[Adjective]	is smelly
	[PP]	is in 2 2
	[NP]	is a pit
died	[]	died
believe	[S]	believe the wumpus is dead

Для того щоб включити механізм субкатегоризації дієслова в граматику, необхідно виконати три етапи. На першому етапі потрібно доповнити категорію VP так, щоб вона приймала параметр субкатегоризації, VP (subcat), який позначає список доповнень, необхідних для формування повного словосполучення VP. Наприклад, дієслово "give" можна перетворити в повне словосполучення VP, додавши субкатегоризацію [NP, PP], словосполучення "give the gold" можна зробити повним, додавши субкатегоризацію [PP], а словосполучення "give the gold to me" вже є повним словосполученням vp, тому його список субкатегоризації є порожнім список, []. На підставі цих міркувань можна скласти наступні правила для словосполучення VP (рис. 1.4).

VP (subcat) → Verb (subcat)
 | VP (subcat + [NP]) NP (Objective)
 | VP (subcat + [Adjective]) Adjective
 | VP (subcat + [PP]) PP

Рисунок 1.4 – Правила для словосполучення VP

Останній рядок цих правил можна прочитати таким чином: "Словосполучення VP із заданим списком субкатегоризації subcat можна сформувати на основі вкладеного словосполучення VP, за яким слідує PP, за умови, що вкладене VP має список субкатегоризацій, який починається з елементів списку subcat і закінчується символом PP". Наприклад, вираз VP ([]) формується за допомогою VP([PP]), за яким слідує PP. А перший рядок цих правил говорить про те, що словосполучення VP зі списком субкатегоризації subcat може бути сформовано за допомогою категорії Verb з тим же самим списком субкатегоризації. Зокрема, вираз VP ([NP]) можна сформувати за допомогою Verb ([NP]). Одним із прикладів такого дієслова є "grab", тому словосполучення "grab the gold" (хапай золото) позначається як VP ([]).

На другому етапі необхідно змінити правило для S так, щоб в ньому було потрібне дієслівне словосполучення, що має при собі всі свої доповнення і тому супроводжується списком субкатегоризацій []. Це означає, що словосполучення "I grab the gold" (Я хапаю золото) являє собою допустиме речення, а "You give" – ні. Таким чином, вступає в силу наступне правило: $S \rightarrow NP(\text{Subjective})VP([])$, яке можна прочитати так: "Пропозиція може складатися зі словосполучення NP в називному відмінку, за яким слідує словосполучення VP, що має порожній список субкатегоризації". На рис. 1.5 показано дерево синтаксичного аналізу речення "You give me the gold", отримане за допомогою цієї граматики.

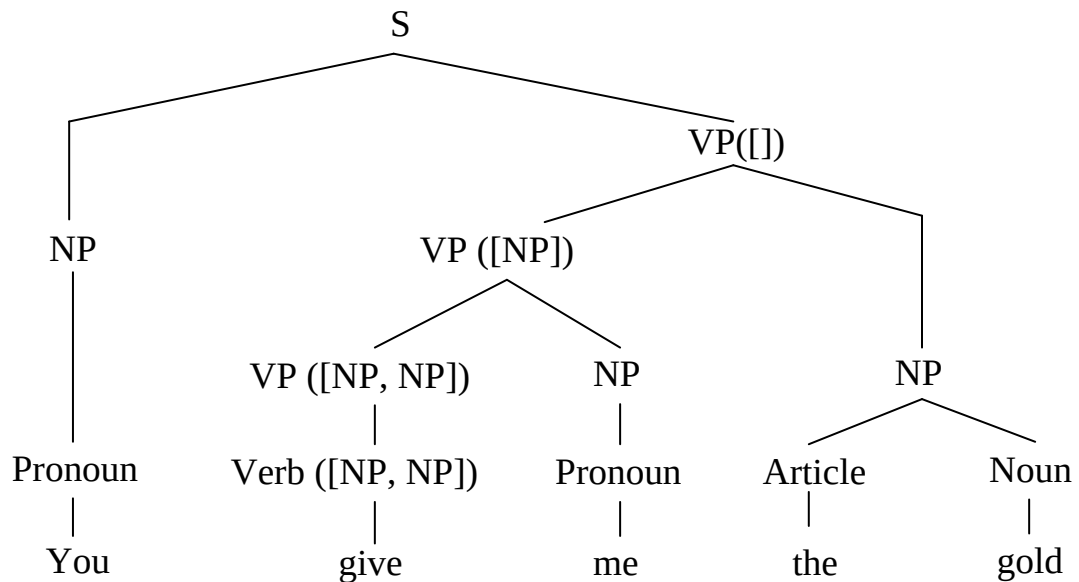


Рисунок 1.5 – Дерево синтаксичного аналізу речення, на прикладі якого показана субкатегоризація дієслова та дієслівного словосполучення

На третьому етапі має бути враховано те, що дієслівні словосполучення (та інші словосполучення), окрім доповнень, можуть також мати додатки, або віддієслівні доповнення, що представляють собою словосполучення, які не пов'язані з окремим дієсловом, а, скоріше, можуть з'являтися у будь-якому дієслівному словосполученні. До додатків відносяться словосполучення, що представляють час і місце, оскільки майже

будь-яка дія або подія може бути пов'язана з певним часом чи місцем. Наприклад, додатками є прислівник "now" в словосполученні "I smell a wumpus now" (Зараз я відчуваю запах вампуса) і запропоноване словосполучення PP "on Tuesday" в реченні "give me the gold on Tuesday" (Віддай мені золото у вівторок). Нижче наведені два правила, які допускають включення прийменникового і обставинного доповнення у будь-яке дієслівне словосполучення VP.

Кожне розширене правило є схемою правила, замінюючи собою безліч правил, по одному для кожної можливої комбінації значень розширених складових. Здатність розширених граматик залежить від кількості таких комбінацій. Якщо їх кількість є кінцевою, то розширена граматика еквівалентна контекстно-вільній граматичній: схема правила може бути замінена окремими контекстно-вільними правилами. Якщо ж кількість значень нескінченна, то розширені граматичні можуть представляти мови, відмінні від контекстно-вільних.

1.4 Семантична інтерпретація

До сих пір в цьому розділі розглядався тільки синтаксичний аналіз мови. У цьому розділі звернемося до семантики – вилучення сенсу фрагментів мови. У цьому розділі в якості мови уявлення використовується логіка першого порядку, тому семантична інтерпретація являє собою процес зв'язування з деяким словосполученням деякого вираження логіки першого порядку. Інтуїтивно ясно, що зміст словосполучення "the wumpus" – це образ великої, волохатої тварини, який може бути представлений в логіці у вигляді логічного терміна Wumpus, а сенсом словосполучення "the wumpus is dead" є логічне вираження $\text{Dead}\{\text{Wumpus1}\}$. У цьому розділі такі інтуїтивні представлення будуть визначені точніше. Розпочнемо з простого прикладу – з правила для опису позицій в решітці: "NP $\rightarrow$ Digit Digit".

Доповнимо дане правило, додавши до кожної складової параметр, який представляє семантику цієї складової. Буде отримано наступне: “NP([x,y]) $\rightarrow$ Digit(x) Digit(y)”.

Це правило говорить, що рядок, який складається з цифри та семантикою x, за якою слідує ще одна цифра з семантикою у, утворює словосполучення NP з семантикою [x, y], яке представляє собою застосоване нами позначення квадрата в решітці – в світі вампуса.

Тепер ми готові записати семантичні доповнення для невеликої частини англійської мови. Почнемо з визначення того, які семантичні представлення бажано пов'язати з тими або іншими словосполученнями. Розглянемо простий приклад речення "John loves Mary" (Джон любить Мері). Словосполучення NP "John" повинно мати в якості його семантичної інтерпретації логічний термін John, а все речення в цілому повинно мати в якості своєї інтерпретації логічне висловлювання Loves (John, Mary). Ці визначення поки не викликають труднощів. Складнощі виникають при аналізі дієслівного словосполучення VP "loves Mary". Семантична інтерпретація цього словосполучення не є ні логічним терміном, ні повним логічним висловлюванням. Інтуїтивно ясно, що слова "loves Mary" представляють собою опис, який може відноситися або не відноситися до конкретної особи (в даному випадку воно відноситься до Джону). Це означає, що словосполучення "loves Mary" є предикатом, який дозволяє отримати повне логічне висловлювання після його об'єднання з терміном, що представляє деяку особу (та особа, яка виражає любов). Використовуючи λ -позначення, можна представити словосполучення "loves Mary" як наступний предикат: “ λx Loves(x, Mary)”.

Тепер необхідно сформулювати правило, яке означає наступне: "Словосполучення NP з семантикою obj, за яким слідує словосполучення VP з семантикою rel, складають висловлювання, семантика якого є результатом застосування rel к obj": $S(rel(obj)) \rightarrow NP(obj) VP (rel)$.

Це правило повідомляє, що семантичною інтерпретацією речення "John loves Mary" є наступна: $(\lambda x \text{ Loves}(x, \text{Mary})) (\text{John})$, яка еквівалентна вираженню $\text{Loves} (\text{John}, \text{Mary})$.

Інша частина семантики безпосередньо впливає з угод, прийнятих нами досі. Оскільки словосполучення VP представлені як предикати, бажано дотримуватися одноманітності і представити дієслова так само, як предикати. Дієслово "loves" буде представлений у вигляді $\lambda y \lambda x \text{ Loves} (x, y)$, тобто у вигляді предиката, який після підстановки параметра Mary повертає предикат $\lambda x \text{ Loves} (x, \text{Mary})$.

У правилі $\text{VP} \rightarrow \text{Verb NP}$ предикат, який представляє собою семантичну інтерпретацію дієслова, застосовується до об'єкта, який є семантичною інтерпретацією словосполучення NP, для отримання семантичної інтерпретації всього словосполучення VP. В кінцевому підсумку буде отримана граматика, яка показана на рис. 1.6, і дерево синтаксичного аналізу, наведене на рис. 1.7.

$$\begin{aligned} \text{S (rel (obj))} &\rightarrow \text{NP (obj) VP (rel)} \\ \text{VP (rel (obj))} &\rightarrow \text{Verb (rel) NP (obj)} \\ \text{NP (obj)} &\rightarrow \text{Name (obj)} \\ \\ \text{Name (John)} &\rightarrow \text{John} \\ \text{Name (Mary)} &\rightarrow \text{Mary} \\ \text{Verb } (\lambda x \lambda y \text{ Loves}(x,y)) &\rightarrow \text{loves} \end{aligned}$$

Рисунок 1.6 – Граматика, яка дозволяє вивести дерево синтаксичного аналізу та семантичну інтерпретацію для речення "John loves Mary" (і у трьох інших реченнях)

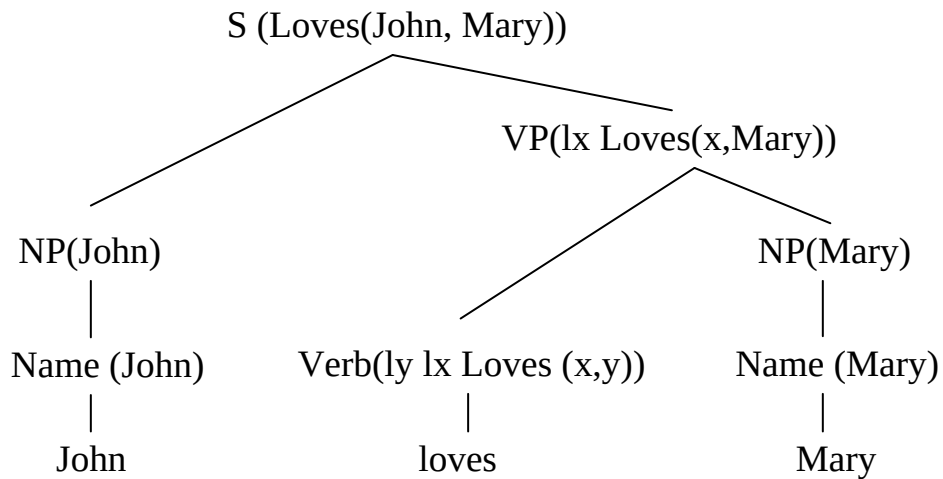


Рисунок 1.7 – Дерево синтаксичного аналізу з семантичними інтерпретаціями для речення "John loves Mary"

Розглянемо речення: "Every agent smells a wumpus" (Кожен агент відчуває запах вампуса). Це речення фактично неоднозначне. Основний його зміст полягає в тому, що навіть якщо вампуси будуть зустрічатися різним агентам, всі вони зуміють відчути їх запах, а ще один варіант смислового трактування цього речення полягає в тому, що існує єдиний вампус, запах якого відчуває кожен агент.

На час відкладемо аналіз цієї проблеми неоднозначності і розглянемо тільки першу інтерпретацію. Спробуємо проаналізувати її композиційно, підрозділяючи все речення на компоненти.

При цьому відразу ж виникають дві складності. Перша проблема полягає в тому, що семантика усього речення здається визначуваною семантикою словосполучення NP, притому що семантика словосполучення VP заповнює тільки частину Р. Це означає, що семантику всього речення неможливо сформувати за допомогою конструкції $rel(obj)$. Але це можна зробити за допомогою конструкції $obj(rel)$, яка виглядає трохи дивно (принаймні на перший погляд). Друга проблема полягає в тому, що необхідно отримати змінну а як параметр відношення $Smell$. Іншими словами, семантика речення формується шляхом вставки семантики словосполучення VP в правильний слот параметра словосполучення NP, і разом з цим вставки

змінної a із словосполучення NP в правильний слот параметра семантики словосполучення VP. Створюється таке враження, що потрібні дві функціональні композиції, тому вся ця задача обіцяє стати досить складною. Але вся складність обумовлена тим фактом, що семантична структура цієї пропозиції значно відрізняється від синтаксичної структури.

Для запобігання подібної плутанини в багатьох сучасних граматиках прийнятий інший підхід. У них визначена деяка тимчасова форма, яка служить в якості посередника між синтаксисом і семантикою. Ця тимчасова форма має дві ключові властивості. По-перше, вона структурно аналогічна синтаксису речення і тому може бути легко сконструйована за допомогою композиційних засобів. По-друге, вона містить достатньо інформації для того, щоб її можна було перевести в звичайне висловлювання логіки першого порядку. Оскільки ця форма займає положення між синтаксичними і логічними формами, її називають квазілогічною формою. В даному розділі буде використовуватися квазілогічна форма, яка включає всі засоби логіки першого порядку і доповнена лямбда-виразами та однієї новою конструкцією, яку ми будемо називати кваліфікованим терміном.

При створенні граматики, що дозволяє виробляти квазілогічну форму, можна залишити незмінними раніше отримані правила. Правило для S все ще створює семантику речення S зі значенням $\text{rel}(\text{obj})$. Але деякі правила дійсно змінюються. Наприклад, граматичне правило для артикля "a" виглядає таким чином: $\text{Article} (E) \rightarrow a$. А правило застосування артикля в поєднанні з іменником є таким: $\text{NP}([qx \text{ sem}(x)]) \rightarrow \text{Article}(q) \text{ Noun}(\text{sem})$.

Це правило говорить про те, що семантика іменного словосполучення NP виражається терміном з квантором q , де квантор позначений артиклем, включає нову змінну x і виражається висловом, сформованим шляхом застосування семантики даного імені до змінної x . Інші правила для словосполучення NP є аналогічними.

Тепер необхідно перетворити квазілогічну форму у форму логіки першого порядку шляхом перетворення термінів з кванторами в справжні

терміни. Таке завдання може бути вирішено за допомогою простого правила – для кожного терміна $[q \times P(x)]$ з квантором q в квазілогічній формі (Quasi – Logical Form – QLF) цей термін з квантором замінити змінною x , а форму QLF замінити виразом $qx P(x)$ or QLF, де or являє собою операцію $\Rightarrow$, якщо q – квантор $\forall$, і операцію $\wedge$, якщо q – квантор $\exists$. Наприклад, речення "Every dog has a day" (У кожної собаки буває свято) має наступну квазілогічну форму: $\exists e((e \in Has([\forall d d \in Dogs], [\exists a a \in Days], Now))$.

Вище було показано, як агент може сприйняти рядок слів і скористатися граматиною для виведення множини можливих семантичних інтерпретацій. А в цьому розділі буде розглядатися проблема створення повної інтерпретації шляхом додавання контекстно-залежної інформації про поточну ситуацію до кожної потенційної інтерпретації.

Найбільш явна потреба в прагматичній інформації виникає при розпізнаванні сенсу вказівних словосполучень, що представляють собою словосполучення, які відносяться безпосередньо до поточної ситуації. Наприклад, у реченні "I am in Boston today" (Я сьогодні перебуваю в Бостоні) інтерпретації вказівних словосполучень "I" і "today" залежать від того, хто і коли вимовив цю пропозицію. Ми представляємо вказівні словосполучення за допомогою "констант" (таких як *Speaker*), а фактично вони є флюентними словосполученнями, тобто залежать від ситуації. Одержувач, який сприймає мовний акт, повинен також сприйняти інформацію про те, ким є відправник, і використовувати цю інформацію для визначення сенсу вказівних словосполучень. Наприклад, одержувач може володіти такими знаннями, що $T((Speaker=Agent_B), Now)$.

Така команда, як "go to [2, 2]", неявно вказує на одержувача. Досі в граматиці, що розглядається нами, для S були визначені тільки розповідні речення. Але цю граматику можна легко доповнити, щоб вона охоплювала також команди.

Команда може бути сформована з словосполучення VP , у якому підмет неявно вказує на одержувача. Необхідно відрізнити команди від

тверджень, тому змінимо правила для S, щоб включити позначення типу мовного акту до складу квазілогічної форми наступним чином: $S(\text{Statement}(\text{Speaker}, \text{rel}(\text{obj}))) \rightarrow \text{NP}(\text{obj}) \text{ VP}(\text{rel})$. $S(\text{Command}(\text{Speaker}, \text{rel}(\text{Hearer}))) \rightarrow \text{VP}(\text{rel})$.

1.5 Неоднозначність та усунення неоднозначності

У деяких випадках одержувачі явно помічають суперечливість деякого мовного фрагмента. Але найчастіше сприймана нами мова здається не логічною. Тому, коли дослідники вперше приступили до вирішення завдання по використанню комп'ютерів для аналізу мови в 1960-х роках, вони були вельми здивовані, дізнавшись про те, що майже будь-який фрагмент промови має високий ступінь неоднозначності, навіть якщо альтернативні його інтерпретації не є очевидними для мовця на рідній мові. Система, в якій визначено великий обсяг граматичних правил і великий словник, може знайти тисячі інтерпретацій для речення, яке зовні здається абсолютно звичайним. Розглянемо речення "The batter hit the ball", яке на перший погляд має несуперечливу інтерпретацію, згідно з якою гравець в бейсбол вдарив по бейсбольному м'ячу. Але буде отримана зовсім інша інтерпретація, якщо сенс, який також міг бути виражений в попередньому реченні, так: "The mad scientist unleashed a tidal wave of cake mix towards the ballroom" (Божевільний учений випустив в танцювальний зал приливну хвилю суміші для кексу). Цей приклад заснований на понятті лексичної неоднозначності, яке описує таку ситуацію, що одне слово може мати більше одного сенсу. Лексична неоднозначність є поширеною проблемою. Слово "back" може бути прислівником ("go back" – йди назад), прикметником ("back door" – задні двері), іменником ("the back of the room" – задня частина кімнати) або дієсловом ("back up your files" – створюй резервні копії своїх файлів). Слово "Jack" може бути власною назвою, іменником (гральна карта, шестикутна металева фішка для гри, морський прапор, риба, самець

мавпи, роз'єм або пристрій для підняття важких предметів) або дієсловом (піднімати домкратом автомобіль, полювати з прожектором або сильно бити по бейсбольному м'ячу).

Синтаксична неоднозначність (що називається також структурною неоднозначністю) може виникати разом з лексичною неоднозначністю або без неї. Наприклад, рядок "I smelled a wumpus in [2,2]" має два варіанти синтаксичного аналізу, в одному з них запропоноване словосполучення "in [2,2]" модифікує іменник (Я відчув запах вампуса, що знаходиться в квадраті [2,2]), а в іншому модифікує дієслово (Перебуваючи в квадраті [2,2], я відчув запах вампуса). Синтаксична неоднозначність призводить до семантичної неоднозначності, оскільки один варіант синтаксичного аналізу означає, що в квадраті [2,2] знаходиться вампус, а інший – що в квадраті [2,2] відчувається його запах. В даному випадку прийняття неправильної інтерпретації за справжню може виявитися смертельно небезпечною помилкою.

Семантична неоднозначність може виявлятися навіть у словосполученнях, що не мають лексичної або синтаксичної неоднозначності. Наприклад, іменне словосполучення "cat person" може відноситися до того, хто любить представників сімейства котячих або є одним з провідних артистів у фільмі "Attack of the Cat People". Словосполучення "coast road" може означати дорогу, яка тягнеться вздовж морського узбережжя або веде до нього.

Нарешті, неоднозначність може бути викликана плутаниною між буквальним і переносним значенням. Фігуральний вислів є важливим елементом поезії, але на подив часто зустрічаються також і в повсякденній мові. Метонімія – це риторичний оборот, в якому один об'єкт використовується для позначення іншого. Наприклад, почувши слова: "Chrysler announced a new model" (Компанія Крайслер оголосила про випуск нової моделі), ми не інтерпретуємо їх як твердження, що компанії можуть говорити, оскільки під цими словами мається на увазі, що оголошення зроблено представником цієї компанії зі зв'язків з громадськістю. Метонімія

зустрічається часто та інтерпретується одержувачами-людьми в основному підсвідомо. На жаль, граматики, представлена в цьому розділі, в своєму безпосередньому вигляді не дозволяє так легко впоратися з цією ситуацією. Щоб врахувати семантику метонімії належним чином, необхідно ввести цілий новий рівень засобів усунення неоднозначності. Для цього потрібно передбачити два об'єкти для семантичної інтерпретації кожного словосполучення в реченні – один для позначення об'єкта, на який буквально посиляється це словосполучення ("Chrysler"), а друге – для метонімічного посилення (представник компанії зі зв'язків з громадськістю). Після цього необхідно сформулювати твердження, що ці два об'єкти пов'язані деяким відношенням.

У ньому йдеться про те, що сутність x , яка дорівнює Chrysler, та сутність, яка оголошує новини від імені компанії, пов'язані метонімічним ставленням. На наступному етапі слід визначити, якого роду метонімічні відносини можуть виникати в реченні. Найпростішим випадком є той, в якому метонімія взагалі відсутня, – буквально вказаний об'єкт x і метонімічний об'єкт.

У прикладі з компанією Chrysler прийнятним узагальненням є те, що назва компанії може використовуватися замість імені представника цієї компанії зі зв'язків з громадськістю.

Інші варіанти метонімії можуть передбачати вказівку імені автора замість його робіт ("I read Shakespeare " – Я читав Шекспіра) або, в більш загальній формі, вказівка назви компанії-виробника замість її продукту (" I drive a Honda " – Я воджу Хонду), а також вказівку частини замість цілого (" The Red Sox need a strong arm " – Для Ред Сокс потрібна тверда рука). Деякі варіанти метонімії, такі як " The ham sandwich on Table 4 wants another beer " (Той бутерброд з шинкою за четвертим столиком вимагає ще одне пиво), є продуктом пізніших змін в мові та інтерпретуються стосовно ситуації.

Правила, описані в даному розділі, дозволяють скласти пояснення для речення " Chrysler announced a new model ", але за самим цим поясненням не

слідуює логічний висновок. Для підготовки потенційних пояснень необхідно використовувати засоби формування імовірнісних або немонотонних міркувань.

Метафора – це фігура мови, в якій словосполучення з одним буквальною значенням використовується в якості вказівки по аналогії на інше значення. Більшість людей вважають, що метафора – це інструмент, який використовується поетами, який не грає великої ролі в повсякденному спілкуванні. Проте цілий ряд метафор застосовується настільки широко, що ми навіть не сприймаємо їх як такі. Однією з таких метафор є представлення про те, що "чим більше, тим вище". Ця метафора дозволяє нам говорити про те, що ціни ростуть, підвищуються або злітають вгору, що температура знижується або падає, що чиясь довіра до когось рухнула або що популярність якоїсь знаменитості різко підскочила або пішла на спад.

Існують два способи, що дозволяють врахувати наявність подібних метафор. Один з них полягає в тому, щоб включити всі знання про метафори в словник – додати нові тлумачення слів "рости", "падати", "підвищуватися" і т.п., та описати їх значення, так як вони відносяться до багатьох слів, а не тільки до висоти. Цей підхід є прийнятним для багатьох додатків, але не охоплює узагальнюючий характер метафори, яка дозволяє людям використовувати нові варіанти її реалізації, такі як "увійти в штопор" або "досягти захмарних висот", без побоювання бути незрозумілими. Другий підхід полягає в тому, що потрібно включити в граматику явно задані знання про загальні метафори і використовувати їх для інтерпретації нових варіантів застосування в міру виявлення в мові. Наприклад, припустимо, що в системі є інформація про метафору "чим більше, тим вище". Іншими словами, система може визначити, що логічні вирази, що посилаються на точку шкали висот, можуть інтерпретуватися як шкала кількості, що вказує на відповідні точки. У такому випадку вираз "sales are high" (обсяги продажів високі) отримає буквально інтерпретацію за типом Altitude (Sales, High), а цей вислів можна буде інтерпретувати метафорично як Quantity (Sales, Much).

1.6 Висновки за розділом 1

Проаналізовано процес обробки текстових даних. Досліджено формальні граматики. Визначено, що формальною граматикою називають спосіб опису формальної мови, тобто виділення деякої підмножини з множини всіх можливих слів деякого кінцевого алфавіту.

Проаналізовано методи синтаксичного аналізу, розширені граматики, семантичну інтерпретацію.

Варто відзначити, що при використанні алгоритму синтаксичного аналізу, що діє зліва направо, доводиться робити припущення про те, чи є перше слово частиною речення, що представляє собою спонукання до дії або питання. В деяких випадках підтвердити правильність таких припущень неможливо (наприклад, для слів "take" або "taken"). Якщо припущення, прийняте в цьому методі, виявляється неправильним, то доводиться здійснювати повне повернення аж до першого слова, що ускладнює процес синтаксичного аналізу та уповільнює його. Такого роду повернення є неминучим, але для підвищення ефективності алгоритму синтаксичного аналізу слід запобігти повторному аналізу деяких словосполучень після кожного повернення.

Тому доцільною є розробка методу синтаксичного аналізу на основі діаграмного підходу, в якому виключено зазначений недолік.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Розробка методу синтаксичного аналізу на основі діаграмного підходу

Як відзначено вище, при використанні алгоритму синтаксичного аналізу, що діє зліва направо, доводиться робити припущення про те, чи є перше слово частиною речення, що представляє собою спонукання до дії або питання. В деяких випадках підтвердити правильність таких припущень неможливо (наприклад, для слів "take" або "taken"). Якщо припущення, прийняте в цьому методі, виявляється неправильним, то доводиться здійснювати повне повернення аж до першого слова, що ускладнює процес синтаксичного аналізу та уповільнює його. Тому доцільною є розробка методу синтаксичного аналізу на основі діаграмного підходу, в якому виключено зазначений недолік

Розглянемо два речення: Have the students in section 2 of Computer Science 101 take the exam; Have the students in section 2 of Computer Science 101 taken the exam? Навіть незважаючи на те, що перші 10 слів в цих реченнях співпадають, вони мають дерева синтаксичного аналізу, що дуже сильно відрізняються один від одного, оскільки перше з них являє собою спонукальне речення, а друге – питальне речення. При використанні алгоритму синтаксичного аналізу, що діє зліва направо, довелося б прийняти припущення про те, чи є перше слово частиною речення, що представляє собою спонукання до дії або питання, але не можна було б підтвердити правильність цього припущення щонайменше до обробки одинадцятого слова, "take" або "taken". Якщо припущення, прийняте в цьому алгоритмі, виявляється неправильним, то доводиться здійснювати повне повернення аж до першого слова. Такого роду повернення є неминучим, але для підвищення ефективності алгоритму синтаксичного аналізу слід запобігти повторному

аналізу словосполучення "the students in section 2 of Computer Science 101" як NP після кожного повернення.

У цьому підрозділі буде розроблений алгоритм синтаксичного аналізу, в якому виключено вказане джерело неефективності. В його основі лежить ідея, що є прикладом динамічного програмування – після проведення аналізу кожного підрядка зберігаються результати, щоб не потрібно було проводити його повторний аналіз в подальшому. Наприклад, виявивши, що словосполучення "the students in section 2 of Computer Science 101" відноситься до типу NP, можна зареєструвати цей результат в структурі даних, відомій під назвою діаграми. Алгоритми, що підтримують ці функції, називаються діаграмними синтаксичними аналізаторами. Оскільки в даному випадку ми маємо справу з контекстно-вільною граматикою, то будь-яке словосполучення, виявлене в контексті однієї гілки простору пошуку, цілком може застосовуватися і у будь-якій іншій гілці пошуку [6].

Діаграма для послідовності з n рядків включає у собі $n+1$ вершину і цілий ряд ребер, які сполучають вершини. На рис. 2.1 показана діаграма з шістьма вершинами (позначеними кружками) і трьома ребрами (позначеними лініями). Наприклад, ребро з міткою $[0, 5, S \rightarrow NP VP \cdot]$ означає, що словосполучення NP, за яким слідує VP, об'єднуються, складаючи вузол S, який охоплює рядок від вершини 0 до вершини 5. Символ $\cdot$ в позначенні ребра 3 відокремлює те, що було знайдено доти, від того, що залишилося знайти. Ребра з символами $\cdot$ в кінці називаються повними ребрами. Наприклад, наступне ребро: $[0, 2, S \rightarrow NP VP \cdot]$ вказує, що словосполучення NP охоплює рядок від вершини 0 до вершини 2 (перші два слова) і якщо можна було знайти словосполучення VP, щоб продовжити це ребро, то отримали б словосполучення S. Ребра, подібні до цього, в яких крапка не досягла кінця, називаються неповними ребрами, і прийнято говорити, що для цього ребра потрібно знайти словосполучення VP.

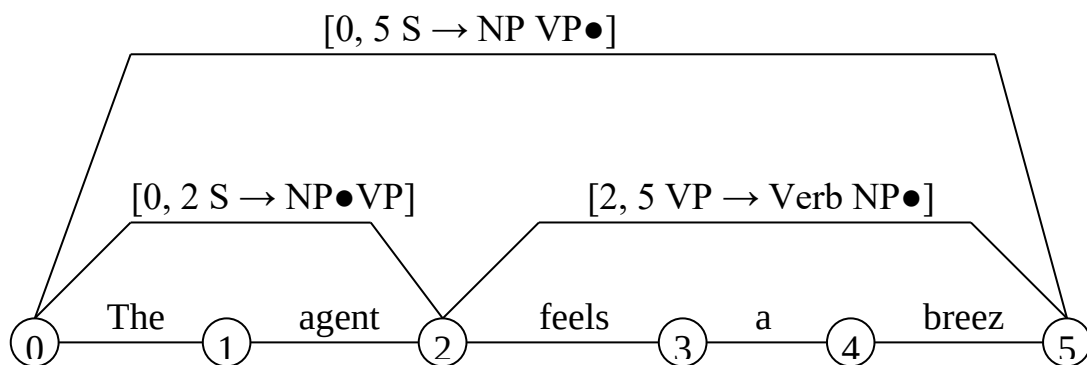


Рисунок 2.1 – Частина діаграми, що відповідає реченню "The agent feels a breeze"

Основна ідея алгоритму діаграмного синтаксичного аналізатора полягає в тому, що в ньому поєднуються найкращі властивості низхідного і висхідного синтаксичного аналізу. Процедура Predictor виконує низхідний аналіз, вона створює записи в діаграмі, які вказують, які символи бажано знайти і в яких місцях. Процедура Scanner – це процедура висхідного аналізу, яка починає роботу зі слів, але використовує будь-яке слово тільки для продовження існуючого запису в діаграмі. По аналогії з нею процедура Extender формує складові дерева синтаксичного аналізу в напрямку знизу вгору, але тільки для продовження існуючого запису в діаграмі.

Після завершення роботи передбачувача (прогнозатору) стосовно до словосполучення алгоритм входить в цикл, в якому викликається процедура Scanner для кожного слова в реченні, що знаходиться у позиції j , є членом такої категорії B , яка відшукується для деякого ребра в позиції j , то дане ребро подовжується, а слово відзначається як екземпляр категорії B . При цьому кожен виклик процедури Scanner призводить до рекурсивного виклику процедур Predictor і Extender, в результаті чого відбувається чергування низхідної та висхідної обробки.

Інший компонент висхідної обробки, процедура Extender, що приймає на вході деяке повне ребро з лівою частиною B і використовує його для продовження будь-якого неповного правила в діаграмі, яке закінчується там,

де починається повне ребро, якщо для цього неповного правила потрібно категорію V.

На рис. 2.1 і в табл. 2.1 показані діаграма і трасування процесу синтаксичного аналізу за допомогою цього алгоритму речення "I feel it" (яке є позитивною відповіддю на питання "Do you feel a breeze?" – "Ви відчуваєте вітерець?").

Таблиця 2.1 – Діаграма синтаксичного аналізу речення "I feel it"

Ребро	Процедура	Етап виведення
a	Initializer	([0, 0, S' → • S])
b	Predictor(a)	([0, 0, S' → • NP VP])
c	Predictor(b)	([0, 0, NP → • Pronoun])
d	Scanner(c)	([0, 1, NP → Pronoun •])
e	Extender(b,d)	([0, 1, S → NP • VP])
f	Predictor(e)	([1, 1, VP → • Verb])
g	Predictor(e)	([1, 1, VP → • VP NP])
h	Scanner(f)	([1, 2, VP → Verb •])
i	Extender(g,h)	([1, 2, VP → VP • NP])
j	Predictor(g)	([2, 2, NP → • Pronoun])
k	Scanner(f)	([2, 3, NP → Pronoun •])
l	Extender(j,k)	([1, 3, VP → VP NP •])
m	Extender(e,l)	([0, 3, S → NP VP •])

У цій діаграмі зареєстровано тринадцять ребер (позначених мітками від a до m), включаючи п'ять повних ребер (показаних над вершинами діаграми) і вісім неповних ребер (показаних під вершинами). В процедурі передбачення Predictor використовується той факт, що для ребра a повинен бути виконаний пошук словосполучення S, щоб можна було дозволити зробити прогноз про те, що повинно з'явитися словосполучення NP (ребро b), а потім словосполучення Pronoun (ребро c). Після цього процедура сканування Scanner розпізнає, що в потрібному місці знаходиться

словосполучення *Pronoun* (ребро *d*), а процедура продовження об'єднує неповне ребро *b* з повним ребром *d* для отримання нового ребра *e*.

Запропонований алгоритм діаграмного синтаксичного аналізатора дозволяє запобігти формуванню великого класу ребер, які довелося б досліджувати при використанні простої висхідної процедури. Розглянемо речення: "The ride the horse gave was wild". Процедура висхідного синтаксичного аналізу відмітила б словосполучення "ride the horse", а потім відкинула б все дерево синтаксичного аналізу, виявивши, що такий варіант аналізу не складається в загальне словосполучення *S*. Але в мові ϵ_0 не дозволяється, щоб словосполучення *VP* йшло за словом "the", тому даний алгоритм діаграмного синтаксичного аналізатора ніколи не передбачив би наявності в цій точці словосполучення *VP* і тому уникнув би марного витрачання часу на формування такого словосполучення *VP*. Алгоритми, які діють зліва направо і запобігають формуванню таких неможливих складових, називаються синтаксичними аналізаторами по лівому куту, оскільки вони формують дерево синтаксичного аналізу, що починається з початкового символу граматики і триває вниз до найлівішого слова в реченні (до лівого кута). Ребро додається до діаграми, тільки якщо воно може послужити для продовження цього дерева синтаксичного аналізу.

Даний діаграмний синтаксичний аналізатор характеризується тільки поліноміальними витратами часу і простору. Він вимагає $O(kn^2)$ простору для зберігання ребер, де *n* – кількість слів у реченні, а *k* – константа, яка залежить від граматики. Якщо алгоритм не може продовжувати подальше формування ребер, він зупиняється, тому відомо, що робота даного алгоритму завжди завершується (навіть за наявності ліворекурсивних правил).

Таким чином, розроблено метод синтаксичного аналізу на основі діаграмного підходу, в якому після проведення аналізу кожного підрядка даних зберігаються результати, щоб що дозволяє не проводити повторний аналіз такого ж словосполучення в подальшому та скоротити час аналізу.

2.2 Вибір інструментарію розробки програмного забезпечення

З метою вибору інструментарію розробки програмного забезпечення виконано порівняльний аналіз різних мов програмування. Результати порівняльного аналізу мов програмування, широко використовуваних для розв'язання складних обчислювальних завдань, наведено у таблиці 2.2.

Таблиця 2.2 – Порівняльний аналіз мов програмування

Критерій порівняння	MATLAB	SciLab	MuPAD
Математичні функції	69,15%	39,74%	51,04%
Стандартна математика	100,00%	80,00%	100,00%
Алгебра	95,29%	76,47%	83,24%
Аналіз	100,00%	100,00%	100,00%
Чисельна математика	80,00%	57,14%	85,71%
Статистика	52,64%	19,09%	20,09%
Інші	61,60%	4,00%	7,60%
Ліцензія	Пропрієтарна	GPL	Пропрієтарна
Робота з графіками	87,19%	48,74%	42,96%
Типи графіків	85,29%	41,18%	38,24%
Експорт/імпорт графіків	92,86%	71,43%	57,14%
Підтримка OLE	+	-	+
C/C++	+	+	+
DLL-виклики	+	+	-

Виходячи з проведеного аналізу, для розробки програми обрано мову Matlab, яка є високорівневою мовою програмування, що інтерпретується, включає засновані на матрицях структури даних, широкий спектр функцій, інтегроване середовище розробки, об'єктно-орієнтовані можливості та інтерфейси до програм, написаних на інших мовах програмування.

Програми, написані на Matlab, бувають двох типів – функції і скрипти. Функції мають вхідні і вихідні аргументи, а також власний робочий простір для зберігання тимчасових результатів обчислень і змінних. Скрипти ж використовують загальний робочий простір. Як скрипти, так і функції не компілюються в машинний код і зберігаються у вигляді текстових файлів. Існує також можливість зберігати так звані pre-parsed програми – функції і

скрипти, оброблені у вид, зручний для машинного виконання. У загальному випадку такі програми виконуються швидше звичайних, особливо якщо функція містить команди побудови графіків.

Основною особливістю мови Matlab є його широкі можливості по роботі з матрицями, які творці мови висловили в гаслі думай векторно.

У складі пакету Matlab є велика кількість функцій для побудови графіків, в тому числі тривимірних, візуального аналізу даних і створення анімованих роликів.

Вбудована середовище розробки дозволяє створювати графічні інтерфейси користувача з різними елементами управління, такими як кнопки, поля введення і іншими. За допомогою компонента Matlab Compiler ці графічні інтерфейси можуть бути перетворені в окремі програми, для запуску яких на інших комп'ютерах необхідна встановлена бібліотека Matlab Component Runtime [10].

Переваги середовища Matlab в порівнянні з іншими середовищами полягають у наявності готових математичних бібліотек, пристосованості роботи з матрицями, зручна візуалізація, легкість роботи з великими об'ємами даних зумовили вибір Matlab в якості інструментарію розробки програмного забезпечення.

2.3 Структурна схема програми

Логічним рішенням під час розробки системи була розробка окремих модулів для реалізації синтаксичного, семантичного та прагматичного аналізу.

Виходячи з цього, було прийнято рішення про розробку наступних модулів: головний модуль MSL.m, модуль синтаксичного аналізу Sint.m, модуль семантичного аналізу Sem.m, модуль прагматичного аналізу Pragm.m.

На основі прийнятого рішення була розроблена структурна схема, зображена на рис.2.2.

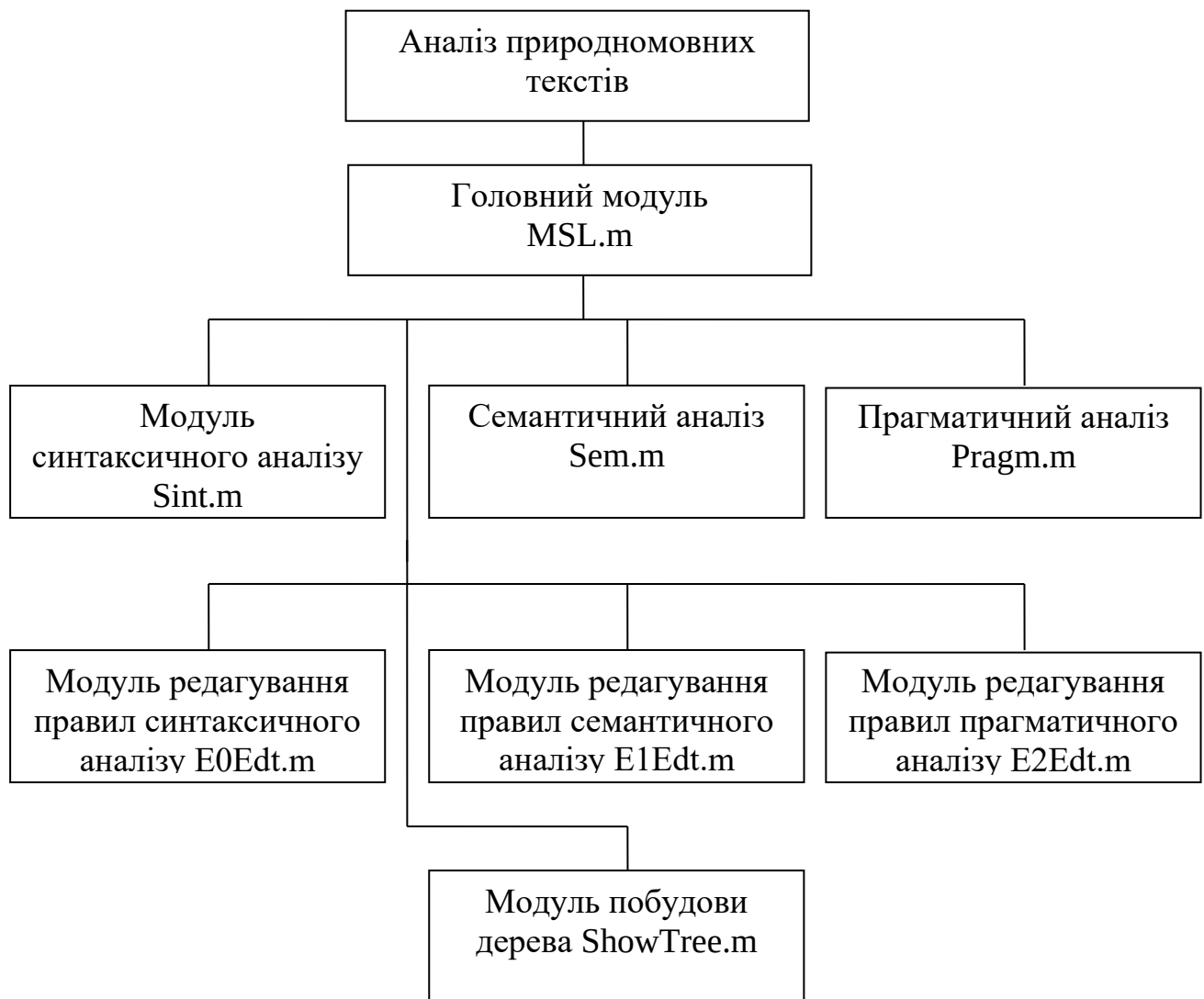


Рисунок 2.2 – Структурна схема програми

2.4 Опис модулів системи, що розробляється

Головний модуль MSL.m є центральним модулем, через який виконується доступ до усіх методів, реалізованих у програмному забезпеченні. У ньому виконується обробка подій, які виникають під час роботи користувача з системою. Реагуючи на натиснення кнопок, система передає далі управління модулю, який відповідає методу, що викликається. Крім того в цьому модулі реалізується завантаження даних з файлу.

Повернені з кожного окремого модуля результати зберігаються у відповідний файл. Також під час роботи відображається поточний стан.

У модулі Sint.m виконується синтаксичний аналіз вибраного речення із застосуванням методів, описаних в розділі 1.

Початковим станом є список слів у вхідному рядку, де кожне із слів розглядається як дерево синтаксичного аналізу тільки з одним листовим вузлом, наприклад [the, wumpus, is, dead].

За допомогою функції визначення наступника виконується пошук в кожній позиції і списку дерев і в кожній правій частині правила грамматики. Якщо підпоследовність списку дерев, що починається з і, узгоджується з правою частиною, то ця підпоследовність замінюється новим деревом, категорією якого є ліва частина правила, а дочірніми вузлами – ця підпоследовність. Під "узгодженням" мається на увазі, що категорія вузла є такою ж, як і елемент в правій частині. Наприклад, правило Article → the узгоджується з підпоследовністю, що складається з першого вузла в списку [the , wumpus , is , dead], тому станом-наступником стає [[Article : the], wumpus , is , dead].

У перевірці мети перевіряється наявність стану, що представляє собою єдине дерево з коренем S.

У модулі Sem.m виконується семантичний аналіз дерева, побудованого на етапі синтаксичного аналізу. На основі спеціального словника встановлюються залежності між словами і будується модифіковане дерево.

У модулі Pragm.m виконується прагматичний аналіз дерева, побудованого на етапі синтаксичного аналізу. На основі словника прагматик встановлюється відповідність між словами з контексту (наприклад, I) і конкретною інформацією.

У модулі E0Edt.m виконується обробка подій, які виникають під час роботи користувача з системою. Реагуючи на натиснення кнопок, система передає далі управління модулю, який відповідає викликаному методу. У

цьому модулі міститься набір функцій для редагування правил синтаксичного аналізу: додавання, видалення та зміна правил.

У модулі E1Edt.m виконується обробка подій, які виникають під час роботи користувача з системою. Реагуючи на натиснення кнопок, система передає далі управління модулю, який відповідає викликаному методу. У цьому модулі міститься набір функцій для редагування правил семантичного розбору: додавання, видалення і зміна правил.

У модулі E2Edt.m виконується обробка подій, які виникають під час роботи користувача з системою. Реагуючи на натиснення кнопок, система передає далі управління модулю, який відповідає викликаному методу. У цьому модулі міститься набір функцій для редагування правил прагматичного розбору: додавання, видалення і зміна правил.

У модулі ShowTree.m виконується графічне відображення дерев, які були сформовані в модулях синтаксичного, семантичного і прагматичного аналізу.

2.5 Висновки за розділом 2

Розроблено метод синтаксичного аналізу на основі діаграмного підходу, в якому після проведення аналізу кожного підрядка даних зберігаються результати, що дозволяє не проводити повторний аналіз такого ж словосполучення в подальшому та скоротити час аналізу.

З метою вибору інструментарію розробки програмного забезпечення виконано порівняльний аналіз різних мов програмування.

Для розробки програмного забезпечення аналізу текстових даних використано мову Matlab, яка містить різні вбудовані засоби для реалізації математичних обчислень та аналізу даних.

Розроблено структурну схему програми, відповідно до якої основними модулями програми є: головний модуль, модуль синтаксичного аналізу, модуль семантичного аналізу, модуль прагматичного аналізу, модуль

редагування правил синтаксичного аналізу, модуль редагування правил семантичного аналізу, модуль редагування правил прагматичного аналізу, модуль побудови дерева аналізу.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Функціональна схема програми

Робота з програмою починається із завантаження текстового файлу в систему.

Після виконаних дій користувач може обрати:

- виконати синтаксичний аналіз;
- виконати семантичний аналіз;
- виконати прагматичний аналіз;
- побудувати дерево;
- змінити правила для побудови дерева синтаксичного аналізу;
- змінити правила для побудови дерева семантичного аналізу;
- змінити правила для побудови дерева прагматичного аналізу;
- завершити роботу з системою.

Кожна модель, реалізована в системі, виділена в окрему функцію.

Функціональна схема розробленої програми представлена на рис. 3.1.

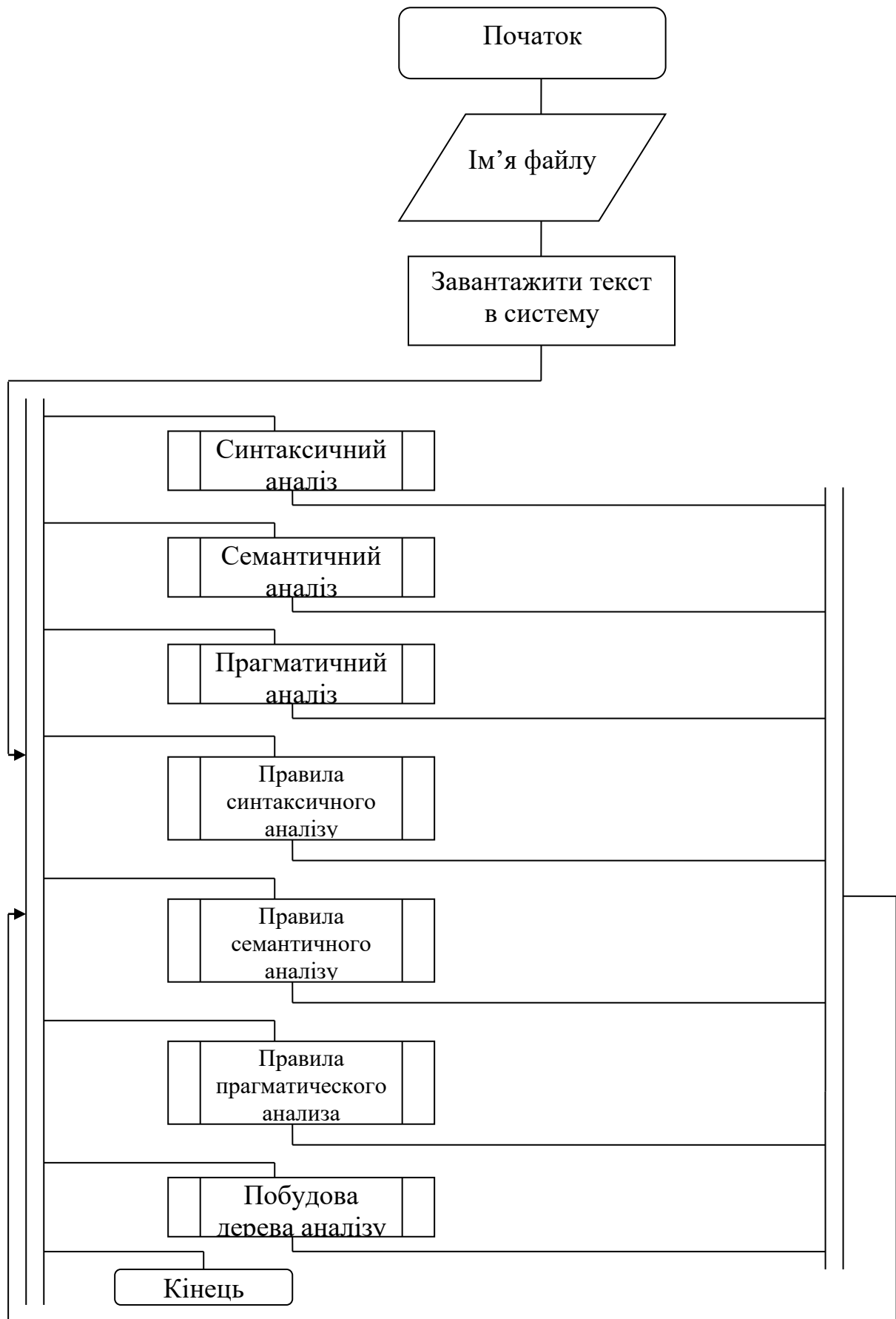


Рисунок 3.1 – Функціональна схема програми

3.2 Експерименти і результати

Для проведення експериментів вибрані тексти англійською мовою. Були вибрані різні речення предметної області, які задовольняють правилам формальної системи предметної області. Загальна кількість речень дорівнює 100.

Для оцінки роботи кожного методу було запропоновано використати порівняння за часом, адже в умовах роботи з великим об'ємом даних це стає досить актуальним завданням. Крім того для оцінки релевантності результатів пропонується оцінити якість результатів під час роботи з системою, хоча це і досить суб'єктивна оцінка, проте саме так проводиться тестування системи інформаційного пошуку для навчання самої системи.

Результати, отримані в ході проведення експериментів за певними критеріями, внесені в таблицю 3.1.

Таблиця 3.1 – Результати експериментів

Метод / Критерій	Висхідний метод	Спадний метод	Метод синтаксичного аналізу на основі діаграмного підходу
Час роботи, с	25,776 с	21,776 с	22,776 с
Рівень точності побудови	Висока. Більшість схем релевантні	Середня. Іноді зустрічаються нерелевантні схеми	Висока. Більшість схем релевантні

В ході проведених експериментів були також отримані дерева речень, аналіз яких виконувався із застосуванням семантичного підходу на основі дерев рішень. Приклад отриманого результату зображений на рис.3.2.

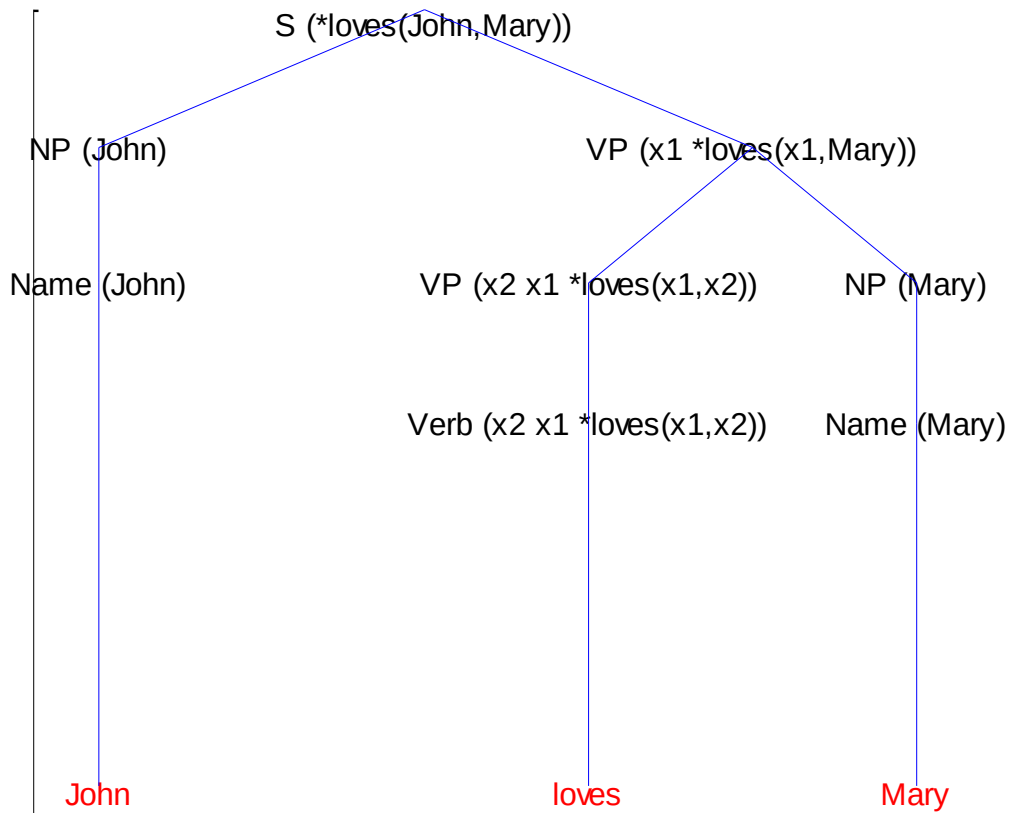


Рисунок 3.2 – Побудоване дерево семантичного аналізу речення

Також були отримані дерева речень, аналіз яких виконувався із застосуванням прагматичного підходу на основі дерев рішень. Приклад отриманого результату зображений на рис.3.3.

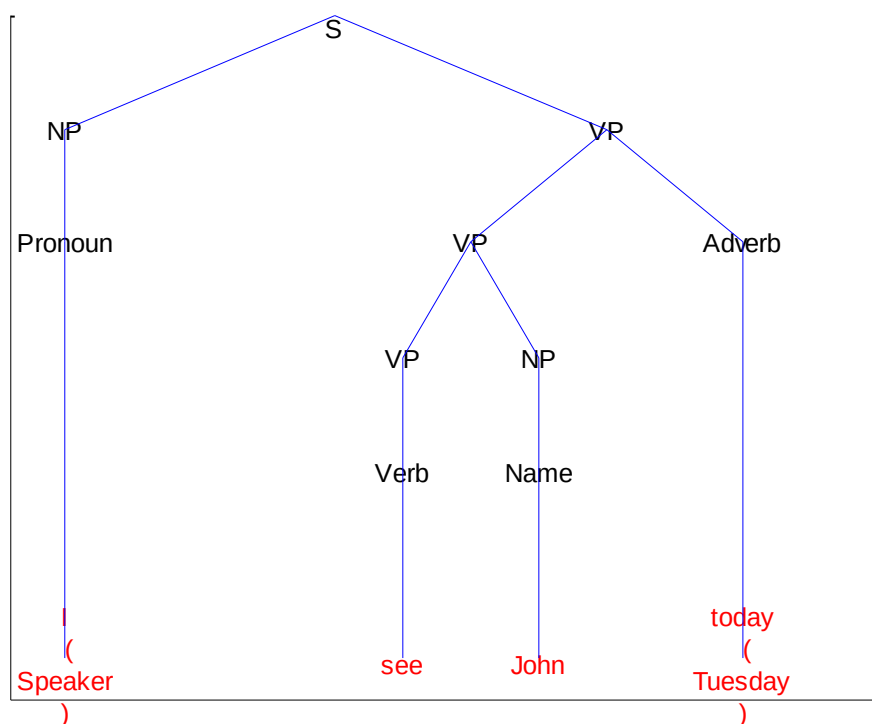


Рисунок 3.3 – Побудоване дерево прагматичного аналізу речення

3.3 Висновки за розділом 3

Запропоновано функціональну схему програми, визначено, що програма дозволяє користувачу виконувати такі функції: синтаксичний аналіз; семантичний аналіз; прагматичний аналіз; побудова дерева аналізу; зміна правил побудови дерева синтаксичного аналізу; зміна правил побудови дерева семантичного аналізу; зміна правил побудови дерева прагматичного аналізу.

Виконано експериментальне дослідження методів аналізу текстових даних. За результатами порівняльного аналізу визначено, що запропонований метод працює швидше у порівнянні із висхідним методом та дозволяє отримувати релевантні результати аналізу текстових даних.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови застосування програми

Програмний продукт призначений для аналізу текстових даних шляхом синтаксичного, семантичного та прагматичного аналізу.

Для коректної роботи програму потрібно запускати в операційній системі Windows 7 або більш пізнішій версії операційної системи.

Програма не вимоглива до процесора та оперативної пам'яті, тому запусниться і швидко працюватиме на будь-якому комп'ютері з операційною системою Windows з використанням мінімуму ресурсів.

Для експлуатації програмного комплексу необхідні такі програмно-технічні засоби:

- ПК під керівництвом операційної системи;
- тактова частота процесора 1 ГГц або більше;
- обсяг оперативної пам'яті не менше 2 Гб;
- обсяг вільного простору на диску – не менше 2000 Мб;
- дозвіл екрана монітора не менше 1024x768;
- маніпулятор миша.

4.2 Характеристики програми

Програма роблена в середовищі Matlab за допомогою графічного інтерфейсу GUI. Для побудови системи використаний модульний підхід.

Робота з системою здійснюється як в графічному режим, так і в командному. Графічний режим інтуїтивно зрозумілий і дозволяє легко ініціювати запуск тих чи інших методів, які реалізовані в системі.

4.3 Звернення до програми

Для звернення до програми необхідно відкрити з операційної системи файл MSL.fig, внаслідок чого буде запущено середовище MATLAB і відкритий графічний інтерфейс. Іншим способом є виконання програми з консолі по команді run ('MSL.m'), але для цього заздалегідь необхідно змінити поточну директорію на ту, в якій зберігається програма, або виклик з консолі окремих модулів та працювати з ними.

4.4 Вхідні і вихідні дані

Вхідними даними для роботи системи виступають: текст, який треба проаналізувати, набір правил формальної системи. Усі вказані вхідні дані знаходяться в окремих файлах, а в програмі вказуються шляхи до файлів, в яких зберігаються ці вхідні дані.

Вихідними даними є синтаксичні, семантичні, а також прагматичні структури проаналізованого тексту, які відображаються графічно.

4.5 Повідомлення

Передача повідомлень про стан роботи інтерфейсного додатку відбувається через поле статусу, яке розташоване в лівому нижньому кутку форми. У ній відображаються наступні повідомлення:

- «selecting file for syntax analysis» – в даний момент відбувається вибір файлу для подальшого аналізу;
- «file selection was canceled» – вибір файлу було скасовано;
- «file for syntax analysis has been loaded» файл для синтаксичного аналізу був успішно завантажений;
- «starting syntax analysis» – виконується синтаксичний аналіз;
- «syntax analysis was finished» – синтаксичний аналіз завершено;

- «no String for syntax analysis» – не вибрано речення для синтаксичного аналізу;
- «starting semantic analysis» – виконується семантичний аналіз;
- «no data from syntax analysis found» – не знайдені результати синтаксичного аналізу;
- «no rules for semantic data» – не знайдені правила для семантичного аналізу;
- «semantic analysis was finished» – семантичний аналіз завершено;
- «starting pragmatic analysis» – виконується прагматичний аналіз;
- «no data from syntax analysis found» – не знайдені результати синтаксичного аналізу;
- «no rules for pragmatic data» – не знайдені правила для прагматичного аналізу;
- «pragmatic analysis was finished» – прагматичний аналіз завершено;
- «starting drawing tree of syntax analysis» – будується дерево синтаксичного аналізу;
- «tree of syntax analysis is drawn» – дерево синтаксичного аналізу побудовано;
- «starting drawing tree of semantic analysis» – будується дерево семантичного аналізу;
- «tree of semantic analysis is drawn» – дерево семантичного аналізу побудовано;
- «starting drawing tree of pragmatic analysis» – будується дерево прагматичного аналізу;
- «tree of pragmatic analysis is drawn» – дерево прагматичного аналізу побудовано.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Програмний продукт призначений для аналізу текстових даних шляхом синтаксичного, семантичного та прагматичного аналізу.

Для коректної роботи програму потрібно запускати в операційній системі Windows 7 або більш пізнішій версії операційної системи.

Інтерфейс користувача забезпечує просту і ефективну взаємодію користувача з програмним продуктом, а також передбачає можливість його адаптації до вимог конкретного користувача.

5.2 Умови виконання програми

Для експлуатації програмного комплексу необхідні такі програмно-технічні засоби:

- ПК під керівництвом операційної системи;
- тактова частота процесора 1 ГГц або більше;
- обсяг оперативної пам'яті не менше 2 Гб;
- обсяг вільного простору на диску – не менше 2000 Мб;
- дозвіл екрана монітора не менше 1024x768;
- маніпулятор миша.

Час роботи програми залежить від конфігурації комп'ютера, на якому вона виконується.

5.3 Виконання програми

Для виконання програми необхідно відкрити з операційної системи файл MSL.fig, внаслідок чого буде запущено середовище MATLAB і відкритий графічний інтерфейс. Іншим способом є виконання програми з консолі по команді run ('MSL.m'), але для цього заздалегідь необхідно

змінити поточну директорію на ту, в якій зберігається програма, або виклик з консолі окремих модулів та працювати з ними.

5.4 Приклад роботи з програмою

Загальний вигляд програми після її запуску зображено на рисунку 5.1.

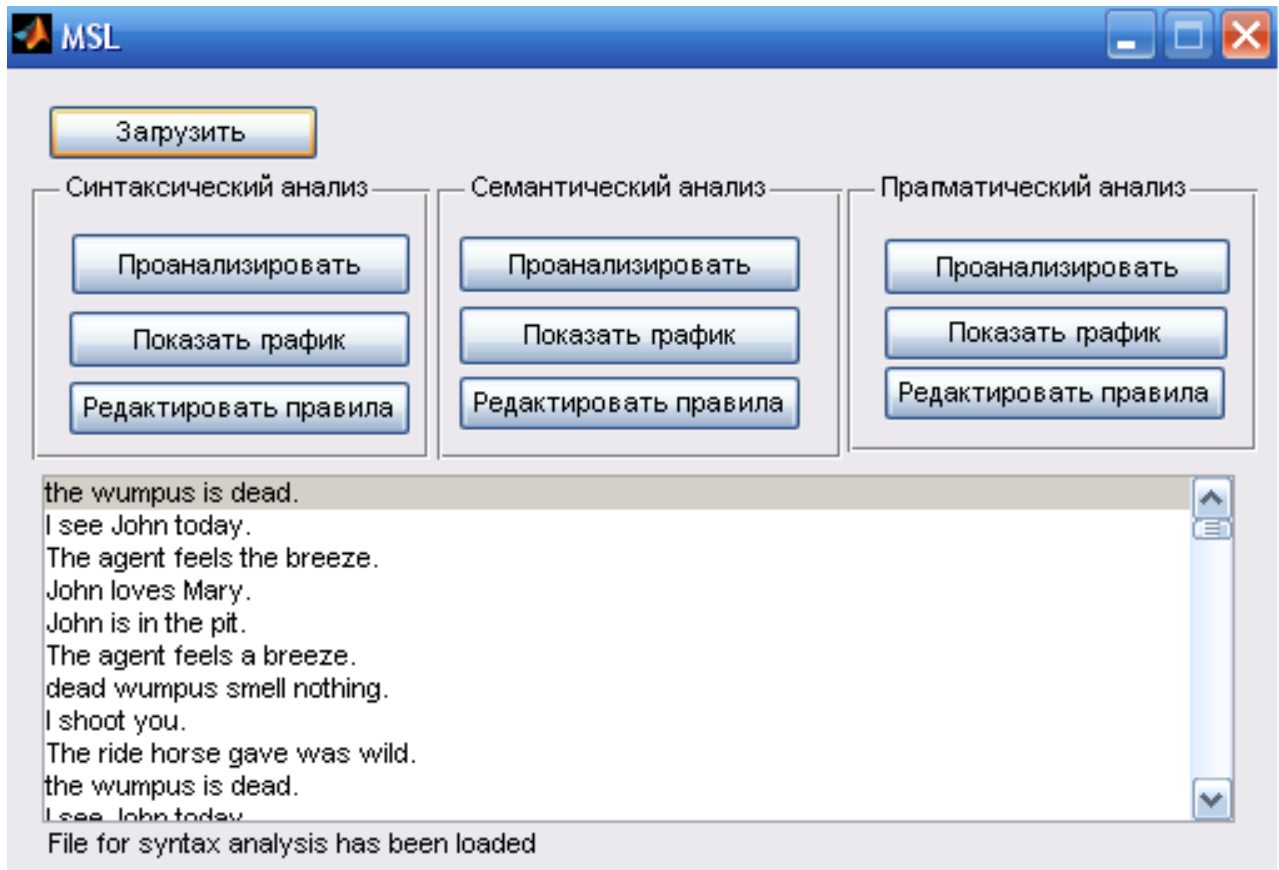


Рисунок 5.1 – Загальний вигляд програми

Після запуску програми необхідно завантажити текстовий файл, в якому міститься текст для аналізу. Для цього необхідно натиснути кнопку «Завантажити». При цьому відкриється зручне діалогове вікно вибору файлів з можливістю переходу між папками (рис. 5.2).

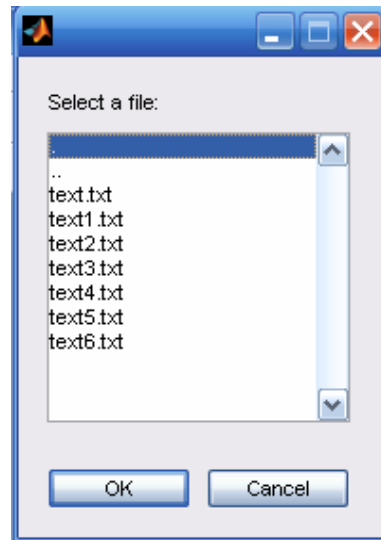


Рисунок 5.2 – Вікно вибору файлу

Далі необхідно виконати синтаксичний аналіз. Про завершення синтаксичного аналізу буде проінформовано в полі статусу. При виконанні синтаксичного аналізу використовуються правила, які можна відредагувати, натиснувши кнопку «Редагувати правила» (рис. 5.3). Далі можна подивитися дерево синтаксичного розбору (рис. 5.4) або виконати семантичний аналіз.

Дії, які необхідні для виконання семантичного та прагматичного аналізу аналогічні діям, описаним вище для синтаксичного аналізу.

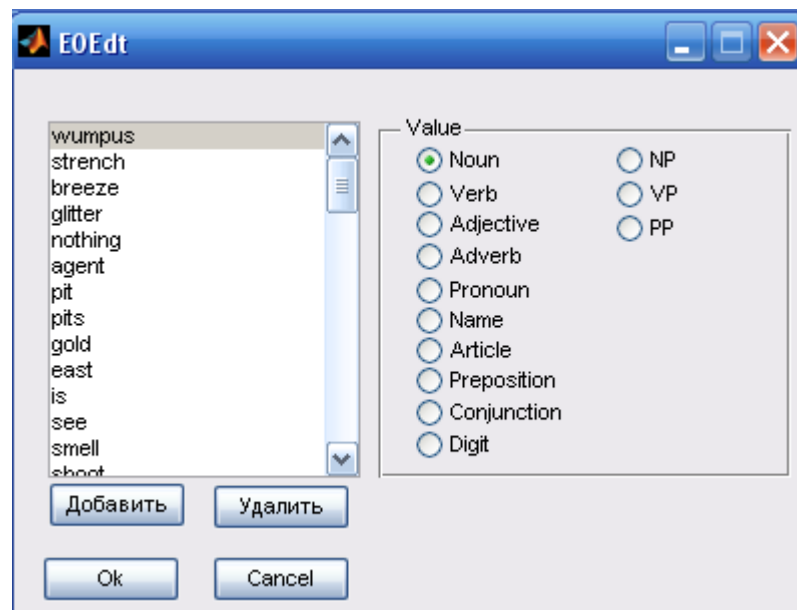


Рисунок 5.3 – Вікно редагування правил синтаксичного розбору

При натисканні на кнопку «Редагувати правила» для синтаксичного аналізу відкриється вікно E0Edt (рис. 5.3). У якому можна переглянути поточне правило. Для цього необхідно вибрати в лівій частині вікна правило, та в правій частині вікна відобразиться значення для обраного правила (NP – іменникове словосполучення, VP – дієслівне словосполучення, PP – прийменникове словосполучення). Для зміни значення обраного правила необхідно вибрати в правій частині вікна нове значення. Для додавання нового правила необхідно натиснути кнопку "Додати". Після чого відкриється вікно, в якому необхідно ввести правило, що додається. Правило, яке містить декілька елементів необхідно розділити пропуском. Це правило буде додано до кінця списку зі значення Noun (іменник). Для видалення обраного правила необхідно натиснути кнопку «Видалити», після чого відповісти згодою на питання про підтвердження видалення. Після роботи з вікном редагування правил можна натиснути кнопку "Ok" для підтвердження внесених змін або кнопку "Cancel" для скасування цих дій.

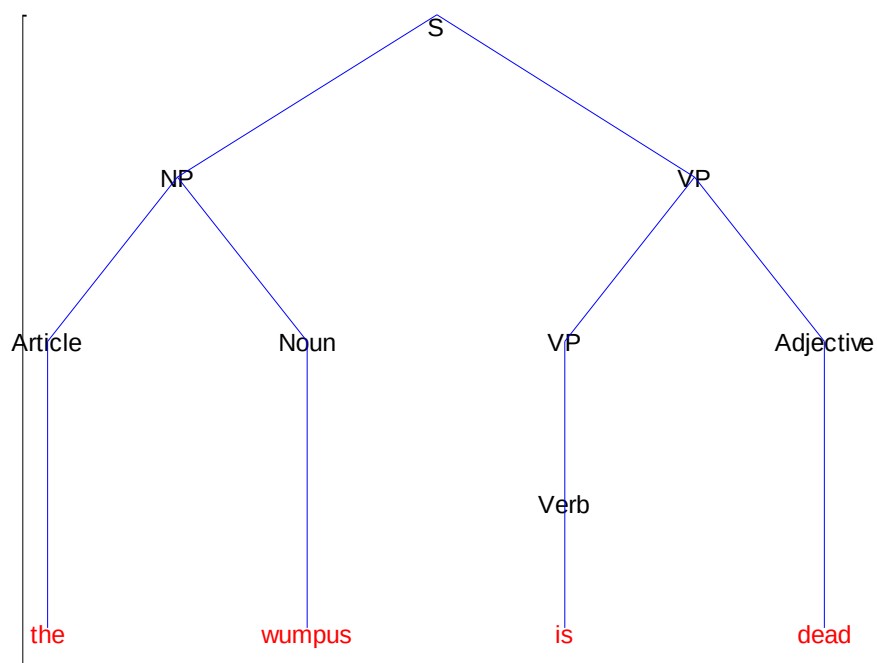


Рисунок 5.4 – Дерево синтаксичного розбору для речення "The wumpus is dead"

Вікно редагування правил семантичного розбору представлене на рис. 5.5, а для прагматичного розбору - на рис. 5.6. Дії, необхідні для редагування правил семантичного і прагматичного аналізу аналогічні діям, необхідним для редагування правил синтаксичного аналізу, описаним вище.

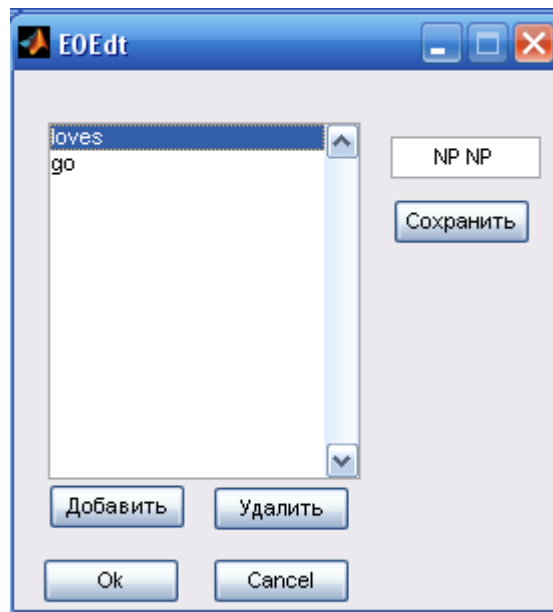


Рисунок 5.5 – Вікно редагування правил семантичного розбору

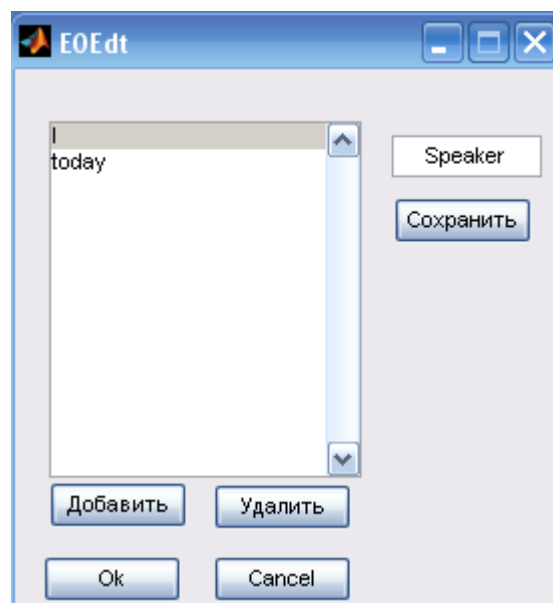


Рисунок 5.6 – Вікно редагування правил прагматичного розбору

5.5 Повідомлення оператору

Передача повідомлень про стан роботи інтерфейсного додатку відбувається через поле статусу, яке розташоване в лівому нижньому кутку форми. У ній відображаються наступні повідомлення:

- «selecting file for syntax analysis» – в даний момент відбувається вибір файлу для подальшого аналізу;
- «file selection was canceled» – вибір файлу було скасовано;
- «file for syntax analysis has been loaded» файл для синтаксичного аналізу був успішно завантажений;
- «starting syntax analysis» – виконується синтаксичний аналіз;
- «syntax analysis was finished» – синтаксичний аналіз завершено;
- «no String for syntax analysis» – не вибрано речення для синтаксичного аналізу;
- «starting semantic analysis» – виконується семантичний аналіз;
- «no data from syntax analysis found» – не знайдені результати синтаксичного аналізу;
- «no rules for semantic data» – не знайдені правила для семантичного аналізу;
- «semantic analysis was finished» – семантичний аналіз завершено;
- «starting pragmatic analysis» – виконується прагматичний аналіз;
- «no data from syntax analysis found» – не знайдені результати синтаксичного аналізу;
- «no rules for pragmatic data» – не знайдені правила для прагматичного аналізу;
- «pragmatic analysis was finished» – прагматичний аналіз завершено;
- «starting drawing tree of syntax analysis» – будується дерево синтаксичного аналізу;

- «tree of syntax analysis is drawn» – дерево синтаксичного аналізу побудовано;
- «starting drawing tree of semantic analysis» – будується дерево семантичного аналізу;
- «tree of semantic analysis is drawn» – дерево семантичного аналізу побудовано;
- «starting drawing tree of pragmatic analysis» – будується дерево прагматичного аналізу;
- «tree of pragmatic analysis is drawn» – дерево прагматичного аналізу побудовано.

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

У дипломній кваліфікаційній роботі магістра досліджено та програмно реалізовано методи аналізу текстових даних.

Очікується, що впровадження в експлуатацію розробленого програмного забезпечення на підприємстві дозволить скоротити 0,5 ставки одного співробітника, що відповідає за обробку та аналіз текстових даних, з заробітною платою 15 тис. гривень на місяць.

У розробці беруть участь програміст із заробітною платою 12000 грн. в місяць і консультант з заробітною платою 8000 грн. у місяць. На розробку проєкту було витрачено 2 місяці. Розробка програми розпочалася першого вересня і повинна бути виконана до кінця жовтня 2020 року.

6.1 Планування розробки програмного продукту

Весь комплекс розробки програмного продукту можна розділити на етапи [14]. Для кожного етапу вказуються трудомісткість, кількість виконавців і тривалість робіт. Інформація з характеристики робіт по розробці програмного забезпечення зведена у таблицю 6.1.

Таблиця 6.1 – Характеристика робіт по розробці програми

Найменування робіт	Трудомісткість		Виконавці	Тривалість, днів
	люд-дн	%		
Аналіз ПЗ	10	14,9	Програміст Консультант	5 5
Визначення вимог до ПП	6	9	Програміст Консультант	3 3
Проектування структури програми	28	41,9	Програміст Консультант	14 14
Розробка схеми функціонування програми	2	3	Програміст	2
Створення програмного коду	10	14,9	Програміст	10
Тестування і налагодження програмного продукту	6	9	Програміст	6
Складання програмної документації	5	7,5	Програміст	5
Разом	67	100	Програміст Консультант	45 22

6.2 Визначення витрат на розробку програми

Для визначення витрат на розробку програми складають калькуляцію кошторисної вартості робіт, що включає статті:

- основна заробітна плата;
- додаткова заробітна плата;
- відрахування на єдиний соціальний внесок (ЄСВ);
- матеріали та комплектуючі вироби;
- витрати на спеціальне обладнання (на оплату машинного часу ЕОМ для написання і налагодження програмних засобів);
- накладні витрати.

Витрати за статтею «Основна заробітна плата» складаються з планового фонду зарплати всіх категорій працівників, зайнятих в розробці програми. Розрахунок зарплати на підставі даних про трудомісткість, представлений в таблиці 6.2.

Таблиця 6.2 – Основна заробітна плата

Посада виконавця	Чисельність, чол	місячний оклад	Кількість місяців роботи	Сума ЗП, грн
Програміст	1	12000	2	24000
Консультант	1	8000	1	8000
Разом	2			32000

Додаткову заробітну плату приймають рівною 10% від основної зарплати. Вона розраховується за формулою (6.1):

$$ЗП_{\text{доп}} = ЗП_{\text{осн}} \cdot 10\% . \quad (6.1)$$

Підставивши величину основної заробітної плати в дану формулу отримаємо: $ЗП_{\text{доп}} = 32000 \cdot 0.1 = 3200$ грн.

Відрахування на єдиний соціальний внесок (ЄСВ) визначають у відсотковому відношенні від сум основної та додаткової зарплати з урахуванням премій і доплат. Відрахування вираховуються за формулою (6.2):

$$ОТ = (ЗП_{\text{осн}} + ЗП_{\text{доп}}) \cdot 0,22 . \quad (6.2)$$

$$ОТ = (32000 + 3200) \cdot 0,22 = 7744 \text{ грн.}$$

Використовується три найменування матеріалів: картридж – 800 грн, флеш-носій – 100 грн., папір – 100 грн. Витрати на матеріали розраховують за формулою (6.3):

$$M = \sum_{i=1}^n (C_i \cdot N_i \cdot (1 + K_{m.z.}) - C_{io} \cdot N_{io}), \quad (6.3)$$

де M – витрати на матеріали, покупні напівфабрикати і комплектуючі вироби, грн.;

$K_{m.z.}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

C_i – ціна і-го найменування матеріалу, напівфабрикату і комплектуючого, грн.;

N_i – потреба в і-му матеріалі, напівфабрикаті і комплектуючому;

C_{io} – ціна зворотних відходів і-го найменування матеріалу, грн;

N_{io} – кількість зворотних відходів і-го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих.

$$C_{io} = 0; N_{io} = 0; K_{m.z.} = 0,05;$$

$$M = (1 + 0,05) \cdot (800 + 100 + 100) = 1050 \text{ грн.}$$

Таким чином, разом витрати на матеріали та комплектуючі вироби становлять 1050 гривень.

У статтю «Витрати на спеціальне обладнання» входять витрати на придбання, транспортування, монтаж і налагодження нестандартного обладнання. Практично, в даному випадку, в цій статті враховуються витрати на оплату машинного часу ЕОМ для написання і налагодження програмних засобів. Для цього необхідно скласти кошторис «витрат на утримання і експлуатацію устаткування», виходячи з якого визначиться вартість одного машино-години роботи ПК, після множення якої на машинний час пішло на написання і налагодження програми отримаємо витрати на оплату машинного часу.

Амортизаційні відрахування визначають за формулою (6.4):

$$A = BB \cdot \frac{H_a}{100}, \quad (6.4)$$

де BB – балансова вартість обчислювальної техніки, грн;

H_a – норма амортизаційних відрахувань на повне оновлення обчислювальної техніки,%. Приймаємо, що техніка повинна працювати 4 роки, тому $H_a = 25\%$.

Балансова вартість обчислювальної техніки становить 40000 грн. Тому амортизаційні відрахування становлять:

$$A = 40000 \cdot 0,25 = 10000 \text{ грн.}$$

Статтю "Експлуатація обладнання" розраховують підсумовуванням витрат на силову електроенергію і допоміжні матеріали. Витрати на електроенергію визначають за формулою (6.5):

$$C_e = N_n \cdot \Phi_{ef} \cdot K_{зч} \cdot K_{зн} \cdot C_e, \quad (6.5)$$

де N_n – номінальна потужність ЕОМ, кВт;

Φ_{ef} – річний ефективний фонд часу роботи ЕОМ, машино–год;

$K_{зч}$ – середній коефіцієнт завантаження за часом;

$K_{зн}$ – коефіцієнт завантаження по потужності;

C_e – ціна одного кВт · год електроенергії, грн ./ (кВт · год).

Номінальна Потужність ЕОМ – 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ ставити 1800 годин. Середні коефіцієнти завантаження за часом та за потужністю рівні відповідно 0,9 та 0,6. Ціна однієї кіловат–години електроенергії ставить 2,11 грн. Отримуємо:

$$C_e = 0,2 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 410,18 \text{ грн.}$$

Приймаємо, що $C_e = 410$ грн.

Витрати за статтею "Заробітна плата обслуговуючих робітників та відрахування на соціальне страхування в інші фонди" визначають за формулою (6.6):

$$ЗП_{обсл} = \Phi ЗП_z \cdot (1 + K_{відрах}) \cdot \frac{t_{обсл}}{\Phi_{ef.обсл}}, \quad (6.6)$$

де $\Phi ЗП_z$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн.;

$K_{відрах}$ – коефіцієнт, що враховує відрахування на соціальне страхування і в інші фонди;

$t_{обсл}$ – час протягом року, необхідне технічне обслуговування ЕОМ, годин / рік. Приймаємо $t_{обсл} = 20$ год.;

$\Phi_{ef.обсл}$ – річний ефективний фонд часу обслуговуючого персоналу, годин / рік. Приймаємо, що річний ефективний фонд часу обслуговуючого персоналу становить 1800 годин

Заробітна плата обслуговуючого персоналу = 9800 грн/місяць.

$$\Phi ЗП_z = 9800 \cdot 12 = 117600 \text{ грн.}$$

$$ЗП_{обсл} = 117600 \cdot (1 + 0,22) \cdot 20 / 1800 = 1594 \text{ грн.}$$

Суму витрат за статтею "Поточний ремонт обладнання" обчислимо як 3% від балансової вартості обладнання, оскільки неможливо заздалегідь визначити, скільки коштів необхідно витратити на придбання запасних частин наявного обладнання. Балансова вартість обчислювальної техніки становить 40000 грн. Тому сума витрат за статтею "Поточний ремонт обладнання" становить: $40000 \cdot 0,03 = 1200$ грн.

До інших витрат за проектом віднесемо придбання середовища розробки Matlab вартістю 30000 гривень, яке необхідно для розробки програмного забезпечення.

Кошторис витрат на утримання і експлуатацію устаткування наведений в таблиці 6.3.

Таблиця 6.3 – Кошторис витрат на утримання і експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація обладнання	10000
Заробітна плата основна і додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	1594
Експлуатація обладнання (крім витрат на поточний ремонт)	410
Поточний ремонт обладнання	1200
Інші витрати	30000
Разом	43204

Витрати на оплату машинного часу ЕОМ для написання і налагодження програмних засобів визначаються за формулою (6.7):

$$C_{mo} = P_{екс} \cdot t_{mo}, \quad (6.7)$$

де C_{mo} – витрати на оплату машинного часу, грн;

$P_{екс}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн. / машино–год.;

t_{mo} – машинний час цифрової ЕОМ для написання і налагодження даного програмного продукту, машино–год.

$$P_{екс} = 43204/1800 = 24 \text{ грн. / машино-год.}$$

Для написання і налагодження даного програмного продукту ЕОМ експлуатується протягом 45 днів в одну зміну по 8 годин, що складає в сумі 360 годин. Таким чином отримуємо суму за статтею «Витрати на спеціальне обладнання»:

$$C_{мо} = 24 \cdot 360 = 8640 \text{ грн.}$$

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т.д.), утримання та експлуатацію будівель. Накладні витрати включаються до вартості розробки програми непрямым шляхом – у відсотках до основної заробітної плати розробників. Для обчислення цієї статті видатків у дипломному проєкті приймемо, що накладні видатки становлять 35% від основної заробітної плати розробників, що складає 32000 грн. Тому сума накладних видатків становить: $32000 \cdot 0,35 = 11200$ грн.

Калькуляція кошторисної вартості робіт з розробки програми наведена в таблиці 6.4.

Таблиця 6.4 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн.
Основна заробітна плата	32000
Додаткова заробітна плата	3200
ЄСВ	7744

Продовження таблиці 6.4

Найменування статей витрат	Сума, грн.
Матеріали і комплектуючі	1050
Витрати на спеціальне обладнання	8640
Накладні витрати	11200
Разом	63834

6.3 Розрахунок економічної ефективності програмного продукту

Річну економію від впровадження програми, що дозволяє знизити витрати машинного часу ЕОМ для рішення певного завдання в порівнянні із програмою, що застосовувалася раніше, визначають за формулою:

$$E = (T_{M_1} - T_{M_2}) \cdot B_{\text{екс.КТЗ}} + E_n^a \cdot \frac{\Phi_{\text{б.КТЗ}} \cdot (T_{M_1} - T_{M_2})}{\Phi_{\text{еф.КТЗ}}} - \frac{S_{\text{рп}}}{T_c} + E_{\text{зн}}, \quad (6.8)$$

де T_{M_1}, T_{M_2} – машинний час, необхідний для розв’язання поставлених завдань, відповідно в старому та у новому варіантах, машино–год./рік;

$B_{\text{екс}}$ – експлуатаційні витрати, що доводяться на 1 год машинного часу ЕОМ, грн/ машино–год.;

E_n^a – нормативний коефіцієнт ефективності капітальних вкладень у засоби автоматизації;

$\Phi_{\text{б.ктз}}$ – балансова вартість ЕОМ (КТЗ), грн.;

$\Phi_{\text{еф.ктз}}$ – річний ефективний фонд часу роботи ЕОМ, машино–год.;

$S_{\text{рп}}$ – сумарні витрати на розробку програми, грн.;

T_c – термін служби впроваджуваної програми до її морального зношування, років. Приймаємо цей термін 5 років;

E_{zn} – економія на заробітній платі.

До впровадження даного ПЗ дані функції виконувались співробітником, що відповідає за обробку рукописних символів, 1800 годин на рік, а після 900 годин на рік (економія складає 0,5 ставки одного співробітника, що відповідає за обробку та аналіз текстових даних, з заробітною платою 15 тис. гривень на місяць).

$$E = (1800 - 900) \cdot 24 + 0,2 \cdot (40000 \cdot (1800 - 900)) / 1800 - 63834 / 5 + 15000 \cdot 0,5 \cdot (1 + 0,22) \cdot 12 = 122633 \text{ грн.}$$

Коефіцієнт ефективності капітальних вкладень E и строк їхньої окупності $T_{ок}$ розраховуються за формулами (6.9) і (6.10):

$$Ef = \frac{E}{S_{pn}}, \quad (6.9)$$

$$T_{ок} = \frac{S_{pn}}{E}, \quad (6.10)$$

$$Ef = 122633 / 63834 = 1,92,$$

$$T_{ок} = 63834 / 122633 = 0,52 \text{ року.}$$

Зведемо отримані техніко–економічні показники в таблицю 6.5.

Таблиця 6.5 – Техніко–економічні показники

Показники	Сума	Одиниця виміру
Витрати на розробку програми	63834	грн.
Річна економія	122633	грн.
Строк окупності програми	0,52	рік

Таким чином, впровадження в експлуатацію розробленого програмного забезпечення на виробничому підприємстві дозволить скоротити 0,5 ставки одного співробітника, що відповідає за обробку та

аналіз текстових даних, з заробітною платою 15 тис. гривень на місяць. Економічні розрахунки показали, що строк окупності програми 0,52 року, що значно менше часу її нормативного строку окупності (5 років). Крім того, річна економія при використанні програми становить 122633 грн. Виходячи з виконаних розрахунків, можна зробити висновок, що розробка даного програмного продукту є економічно доцільною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

При розробці даного проекту на розробника постійно впливають або періодично діють наступні шкідливі та небезпечні фактори:

- підвищена температура поверхонь ПК;
- підвищена або знижена температура повітря робочої зони;
- виділення в повітря робочої зони ряду хімічних речовин;
- підвищена або знижена вологість повітря;
- підвищений чи знижений рівень негативних і позитивних аероіонів;
- підвищене значення напруги в електричному ланцюзі, замикання;
- підвищений рівень статичної електрики;
- підвищений рівень електромагнітних випромінювань;
- підвищена напруженість електричного поля;
- відсутність або нестача природного світла;
- недостатня штучна освітленість робочої зони;
- підвищена яскравість світла;
- підвищена контрастність;
- прямий і відбитий блискіт;
- зорова напруга;
- монотонність трудового процесу;
- нервово-емоційні перевантаження.

Робота на ПК супроводжується постійним і значним напруженням функцій зорового аналізатора. Однією з основних особливостей є інший принцип читання інформації, ніж при звичайному читанні. При звичайному читанні текст на папері, розташований горизонтально на столі, зчитується працівником з нахиленою головою при падінні світлового потоку на текст. При роботі на ПК оператор зчитує текст, майже не нахиляючи голову, очі дивляться прямо або майже прямо вперед, текст (джерело люмінесцируючих

речовин екрану) формується по інший бік екрану, тому користувач не зчитує відбитий текст, а дивиться безпосередньо на джерело світла, що змушує очі і орган зору в цілому працювати в невластивому йому стресовому режимі тривалий час.

7.2 Заходи щодо забезпечення безпеки

Так як в приміщенні обчислювального центру відносна вологість повітря не перевищує 60%, повітря не містить пил і має нормальну температуру (менше 30° С), а також підлога має ізольоване покриття, то це приміщення згідно класифікації приміщень за ступенем небезпеки ураження електричним струмом ПУЕ «Правил улаштування електроустановок» відноситься до приміщень без підвищеної небезпеки [15].

Заходи щодо забезпечення електробезпеки розроблено відповідно до ДСТУ Б В.2.5-82:2016 «Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом». Основними заходами щодо захисту від ураження електричним струмом є [16]:

- забезпечення недоступності струмоведучих частин, які знаходяться під напругою для випадкового дотику;
- електричне розділення мережі;
- усунення небезпеки ураження при виникненні напруги на корпусах та інших частинах електроустаткування, що досягається захисним заземленням, зануленням, захисним відключенням;
- застосування малих напруг;
- захист від випадкового дотику до струмоведучих частин, застосування огорож, подвійної ізоляції;
- захист від небезпеки при переході напруги;
- контроль і профілактика ушкоджень ізоляції;
- застосування спеціальних електрозахисних засобів — переносних пристроїв і запобіжних заходів;

– організація безпечної експлуатації електроустановок.

Для електроживлення ЕОМ і периферійних пристроїв використовується змінне напруга 220 В. При цьому використано заземлення корпусів електричних пристроїв, яке виконується за допомогою додатково третього проводу в кабелях електроживлення обчислювальної техніки. Для живлення обчислювальної техніки офісне приміщення обладнане спеціальними розетками із заземлюючим контактом, які заземлюються через загальну систему заземлення споруди. Опір заземлювальних пристроїв для електроустановок з напругою до 1000 В, до яких і належить обчислювальна техніка, у відповідність до ПУЕ, складає не більше 4 Ом.

Комп'ютери мають власні блоки живлення, що перетворюють змінний струм мережі у постійний (5 В, 12 В). Потужність даних блоків живлення: 400 – 450 Вт. ПК у приміщенні відносяться до категорії І захисту від ураження електричним струмом, оскільки усі прилади при підключенні до електричної мережі яка заземляється, використовують систему TN. Тобто систему, у якій живильні мережі мають глухе заземлення однієї точки струмопровідних частин джерела живлення, а електроприймачі і відкриті провідні частини електроустановки приєднуються до цієї точки за допомогою відповідно нейтрального і захисного провідників. Корпуси всіх електричних пристроїв виготовляються з неструмопровідних матеріалів, а живлення здійснюється спеціальним кабелем, так щоб виключити ураження людини електричним струмом. У приміщенні електропроводка трифазна, чотирьохжильна, з подвійною ізоляцією. Всі кабелі сховані в коробах. Штепсельні розетки встановлені на висоті одного та 0,2 метри від підлоги. Вимикачі на стінах розташовані на висоті 1,2 метра від підлоги [17].

7.3 Заходи щодо виробничої санітарії та гігієни праці

Заходи щодо забезпечення виробничої санітарії та гігієни праці для обчислювального центру обладнаного ПК з ВДТ розроблені відповідно до вимог ДСанПіН 3.3.2.007-98 «Державні стандартні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [18].

Тип роботи програмістів в обчислювальному центрі належить до операторської роботи, даний тип роботи відносить до категорії 1а, тобто до неї належать роботи, що виконуються сидячи і не потребують фізичного навантаження, робоче місце є постійним, робота проводиться у холодний та теплий періоди року, тому рекомендується підтримувати оптимальні показники мікроклімату.

Оптимальний рівень параметрів виробничого повітряного середовища відповідає ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» (табл. 7.1) [19].

Таблиця 7.1 – Норми мікроклімату в приміщенні

Період року	Категорія робіт	Температура повітря, °С	Відносна вологість повітря, %	Швидкість Руху повітря, м/с
Холодний	Легка	22–24	40–60	0,1
Теплий	Легка	23–25	40–60	0,1-0,2

В обчислювальному центрі передбачено: устрій системи водяного опалення приміщення для забезпечення необхідної температури повітря в холодний період року відповідно ДБН В.2.5-67:2013 «Опалення, вентиляція та кондиціювання» [16].

Проведені дослідження мікрокліматичних умов на комп'ютеризованих робочих місцях показали, що зимою значення відносної вологості повітря часто можуть нижчими за встановлені норми і становити в середньому 30–40%. Це призводить не лише до надмірного висихання слизових оболонок

очей, носа, горла, а й до нагромадження зарядів статичної електрики, що утворюються в процесі роботи комп'ютера. Для усунення таких проблем, в приміщенні передбачена наявність 3 зволожувачів повітря, які допомагають підняти вологість до 60%. Кількість зволожувачів зумовлена тим, що один такий апарат розраховано на приміщення площею до 15 м².

У теплий період року для підтримки нормальної температури повітря в приміщенні використовуються кондиціонери типу «спліт-система». Два кондиціонера розташовані у протилежних частинах обчислювального центру дозволяють підтримувати температуру повітря не вище 24°C. Кількість кондиціонерів зумовлена тим, що один такий апарат розраховано на кондиціювання приміщення площею до 18 м².

Вимоги до освітлення регламентовані ДБН В.2.5-28-2018 «Природне та штучне освітлення» [20], згідно з якими освітлення у приміщеннях з ЕОМ суміщене. Природне освітлення бічне, одностороннє.

В обчислювальному центрі коефіцієнт природної освітленості (КПО) становить в середньому 1,1% (1,2% – у сонячний день та 1% – у хмарну погоду), що відповідає вимогам ДБН В.2.5-28-2018. Робочі місця розташовано так, щоб у поле зору не потрапляли вікна або світлі поверхні світильників.

Найбільш прийнятними для приміщень, де працюють ПК, є люмінесцентні лампи ЛБ (білого світла) і ЛТБ (тепло-білого світла) потужністю 20, 40 або 80 Вт.

Розрахуємо кількість ламп необхідних для освітлення приміщення розміром 9х4х3,5 м освітленістю $E_p = 400$ лк. Коефіцієнт відбиття стелі 70% і стін 50%. Для освітлення використовуються люмінесцентні лампи типу ЛБ в світильниках ЛПО.

Знаходимо індекс приміщення

$$i = \frac{A * B}{(3,5 - 0,7) * (9 + 4)} = \frac{9 * 4}{2,8 * (9 + 4)} = \frac{36}{36,4} = 0,98.$$

Приймаємо коефіцієнт запасу $k = 1,6$ і коефіцієнт нерівномірності освітлення

$$z = E_{cp} / E_{min} = 1,1.$$

При індексі $i=0,98$ отримуємо $\eta=43\%$.

Світильники розміщуємо в два ряди ($N_p = 2$)

Визначаємо необхідний світловий потік ламп в кожному ряду:

$$\Phi_{\Sigma} = \frac{E_n \cdot S \cdot k_z \cdot z}{N_p \cdot \eta} = \frac{400 \cdot 36 \cdot 1,1 \cdot 1,6}{2 \cdot 0,43} = 34267.$$

Якщо в світильнику встановити по дві лампи ЛБ ($n = 2$) потужністю 80 Вт і світловим потоком $\Phi_{л} = 5400$ лм, то необхідне число світильників в ряду складе

$$N_{с.р.} = \frac{\Phi_p}{\eta \cdot \Phi_{л}} = \frac{34267}{2 \cdot 5400} = 3,17 = 4 \text{ шт.}$$

Під впливом шуму знижується концентрація уваги, порушуються фізіологічні функції, з'являється стомленість у зв'язку з підвищеними енергетичними витратами і нервово-психічною напругою, погіршується мовна комутація.

Згідно із технічним паспортом обладнання (ПК, монітори, принтери, клавіатури (при натисканні), маніпулятори типу «миша» (при натисканні)), що використовується для роботи, рівень шуму складає: 43 дБА. Згідно із ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвук» нормоване значення рівня шуму більше, ніж фактичне, тож додаткових робіт для усунення надмірного шуму не проводилося. Роботи в даному приміщенні з таким обладнанням відповідають нормативним вимогам [21].

Обладнання і організація робочого місця в обчислювальному центрі мають забезпечувати відповідність конструкції всіх елементів робочого місця та їх взаємного розташування ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності.

Згідно НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроям» [22] площа на одне робоче місце з ПК складає по 6 м², а об'єм – по 20 м³. Приміщення має природне та штучне освітлення. Робочі місця розташовано так, щоб природне світло падало збоку.

Робочі місця та взаємне розташування всіх його елементів відповідає антропометричним, фізичним і психологічним вимогам.

Робочі стільці мають підйомно-повторним механізми та регулюються по висоті та куту нахилу сидіння і спинки, а також по відстані спинки від переднього краю сидіння. Регулювання кожного параметра незалежне, легко здійснюється та має надійну фіксацію. Екран монітору знаходиться на оптимальній відстані 700 мм.

Також приміщення обладнано шафами для зберігання документів, магнітних дисків, полицями, стелажми та тумбами.

Колір приміщень і меблів сприяє створенню комфортних умов для зорового сприйняття.

Полімерні матеріали, використовувані для внутрішнього оздоблення інтер'єру приміщень, дозволені для застосування органами і установами Державного санітарно-епідеміологічного нагляду. Поверхня підлоги в приміщеннях експлуатації ПК рівна, без вибоїн, неслизька, зручна для очищення та вологого прибирання, має антистатичні властивості.

Працюючі з ВДТ і ПЕОМ проходять обов'язкові медичні огляди : попередній - при влаштуванні на роботу і періодичний - протягом трудової діяльності відповідно до наказу МОЗ України № 45 від 31.03.94 р.

7.4 Заходи безпеки у надзвичайних ситуаціях

7.4.1 Заходи з пожежної безпеки

Обчислювальний центр, згідно ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» [23] відноситься до категорії «П-Па», а клас можливої пожежі, згідно ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [24], визначається, як «А».

Оскільки приміщення обчислювального центру має площу 36 м², тому відповідно до вимог п. 3.8 розділу «Типові норми належності вогнегасників» ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги» [25] для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 2 штук (з розрахунку один вогнегасник с величиною заряду вогнегасної речовини 3 кг. і більше, на 20 м² площі приміщення). Відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м.

Усі працівники обчислювального центру під час прийняття на роботу і в процесі праці проходять протипожежний інструктаж, перевірку знань з питань пожежної безпеки.

Евакуаційні виходи і проходи утримуються постійно вільними та нічим не захащуються.

Відповідальний за протипожежний стан приміщень обчислювального центру періодично (не рідше одного разу на місяць) перевіряє працездатність пристрою, який забезпечує автоматичне вимкнення системи вентиляції у разі пожежі, а також вогнедимозатримувальних пристроїв. Під час ремонту слід не допускати руйнування або порушення цілісності негорючих діафрагм у фальш-підлоговому просторі.

Фільтри припливно-витяжної вентиляції слід чистити згідно з затвердженим графіком, але не рідше двох разів на рік. Роботи з ремонту вузлів (блоків) ЕОМ, як правило, проводити не дозволяється. Їх необхідно

проводити в окремому приміщенні (майстерні). У разі необхідності проведення ремонту або технічного обслуговування ЕОМ безпосередньо в залі допускається мати не більше 0,5 л легкозаймистих рідин у небиткій, щільно закритій тарі. Для промивання деталей необхідно застосовувати, як правило, негорючі миючі препарати. Чарунки та інші знімні пристрої допускається промивати горючими рідинами тільки в спеціальному приміщенні, обладнаному припливно-витяжною вентиляцією.

Відповідальний за пожежну безпеку приміщень обчислювального центру зобов'язаний негайно сповістити технічну службу у разі виявлення незадовільного технічного стану димових та теплових сповіщувачів автоматичної пожежної сигналізації, а також про несправності автоматичної установки об'ємного (газового) пожежогасіння.

Усі працівники обчислювального центру повинні вміти привести в дію (в разі потреби) первинні засоби пожежогасіння. Регулярно, але не рідше одного разу на квартал, слід очищати від пилу ЕОМ, контрольновимірну апаратуру, кабельні траншеї та фальшпідлоговий простір за допомогою пиловсмоктувача.

7.4.2 Заходи з цивільного захисту

Евакуації підлягає населення, яке проживає в населених пунктах, що знаходяться у зонах можливого катастрофічного затоплення, можливого небезпечного радіоактивного забруднення, хімічного ураження, в районах виникнення стихійного лиха, аварій і катастроф (якщо виникає безпосередня загроза життю та здоров'ю людей).

Залежно від обстановки, яка склалася на час надзвичайної ситуації техногенного та природного характеру, може бути проведено загальну або часткову евакуацію населення тимчасового або безповоротного характеру.

Загальна евакуація проводиться за рішенням Кабінету Міністрів України для всіх категорій населення і планується на випадок:

- можливого небезпечного радіоактивного забруднення територій навколо атомних електростанцій (якщо виникає безпосередня загроза життю та здоров'ю людей, які проживають в зоні ураження);

- виникнення загрози катастрофічного затоплення місцевості з чотиригодинним добіганням проривної хвилі.

Часткова евакуація проводиться за рішенням Кабінету Міністрів України у разі загрози або виникнення надзвичайної ситуації техногенного та природного характеру. Евакуаційні заходи здійснюються Радою міністрів Автономної Республіки Крим, місцевими органами виконавчої влади, органами місцевого самоврядування.

В даній роботі буде розглядатись такий вид евакуації, як евакуація людей з приміщень і будівель.

Проведення евакуації з приміщень і будівель. Проведення організованої евакуації з виробничих та інших приміщень і будівель, запобігання проявам паніки і недопущення загибелі людей забезпечується шляхом:

- планування евакуації людей (складання плану евакуації з приміщення);

- визначення зон, придатних для розміщення евакуйованих з потенційно небезпечних зон;

- організації оповіщення керівників підприємств і людей про початок евакуації;

- організації управління евакуацією;

- навчання населення діям під час проведення евакуації.

Вимоги до будівель і споруд в контексті евакуації. До всіх будівель і споруд має бути забезпечений вільний доступ. Протипожежні розриви між будинками, спорудами, відкритими майданчиками

- для зберігання матеріалів, устаткування тощо повинні відповідати вимогам будівельних норм. Їх не дозволяється захаращувати, використовувати для

– складування матеріалів, устаткування, стоянок транспорту, будівництва та встановлення тимчасових будинків і споруд, у тому числі інвентарних побутових приміщень, індивідуальних гаражів тощо.

Територія підприємств та інших об'єктів повинна мати зовнішнє освітлення, яке забезпечує швидке знаходження пожежних драбин, протипожежного обладнання, евакуаційних виходів будинків та споруд.

На території промислової будівлі чи споруди на видних місцях повинні бути розміщені плани евакуації, встановлені таблички із зазначенням порядку виклику пожежної охорони, знаки місць розміщення первинних засобів пожежогасіння.

У разі перепланування приміщень, зміни їх функціонального призначення, застосування нового технологічного устаткування необхідно дотримуватися протипожежних вимог чинних нормативних документів будівельного та технологічного проектування. Не дозволяється зниження проектних меж вогнестійкості конструкцій та погіршення умов евакуації людей.

У разі необхідності встановлення на вікнах приміщень, де перебувають люди, ґратів, останні повинні розкриватися, розсуватися або зніматися. Під час перебування в цих приміщеннях людей ґрати мають бути відчинені (зняті).

Під час виробничих приміщень з масовим перебуванням людей слід дотримуватись таких вимог - при кількості людей понад 50 осіб використовувати приміщення, забезпечені не менше ніж двома евакуаційними виходами, що відповідають вимогам будівельних норм, не мають на вікнах глухих ґрат і розташовані не вище другого поверху в будівлях з горючими перекриттями.

Евакуаційні шляхи і виходи повинні утримуватися вільними, нічим не захащуватися і в разі виникнення надзвичайної ситуації забезпечувати безпеку під час евакуації всіх людей, які перебувають у приміщеннях будівель та споруд.

Кількість та розміри евакуаційних виходів з будівель і приміщень, їхні конструктивні й планувальні рішення, умови освітленості, забезпечення незадимленості, протяжність шляхів евакуації, їх облицювання (оздоблення) повинні відповідати протипожежним вимогам будівельних норм.

У разі розміщення технологічного, експозиційного та іншого обладнання в приміщеннях повинні бути забезпечені евакуаційні проходи до сходових кліток та інших шляхів евакуації відповідно до будівельних норм.

У приміщенні, яке має один евакуаційний вихід, дозволяється одночасно розміщати (дозволяється перебування) не більше 50 осіб.

Двері на шляхах евакуації повинні відчинятися в напрямку виходу з будівель (приміщень).

Допускається влаштування дверей з відчиненням усередину приміщення в разі одночасного перебування в ньому не більше 15 чоловік, а також у санвузлах, з балконів, лоджій, площадок зовнішніх евакуаційних сходів (за винятком дверей, що ведуть у повітряну зону незадимлюваної сходової клітки).

При наявності людей у приміщенні двері евакуаційних виходів можуть замикатися лише на внутрішні запори, які легко відмикаються.

Сходові марші і площадки повинні мати справні огорожі з поручнями, котрі не повинні зменшувати встановлену будівельними нормами ширину сходових маршів і площадок.

Сходові клітки, внутрішні відкриті та зовнішні сходи, коридори, проходи та інші шляхи евакуації мають бути забезпечені евакуаційним освітленням відповідно до вимог будівельних норм та правил улаштування електроустановок. Світильники евакуаційного освітлення повинні вмикатися з настанням сутінків у разі перебування в будівлі людей.

Шляхи евакуації, що не мають природного освітлення, повинні постійно освітлюватися електричним світлом (у разі наявності людей) [26].

ВИСНОВКИ

У ході виконання дипломної кваліфікаційної роботи магістра було розв'язано актуальне завдання дослідження та програмної реалізації методів аналізу текстових даних. Проаналізовано процес обробки текстових даних. Досліджено формальні граматики. Визначено, що формальною граматикою називають спосіб опису формальної мови, тобто виділення деякої підмножини з множини всіх можливих слів деякого кінцевого алфавіту. Проаналізовано методи синтаксичного аналізу, розширені граматики, семантичну інтерпретацію.

З метою вибору інструментарію розробки програмного забезпечення виконано порівняльний аналіз різних мов програмування. Для розробки програмного забезпечення аналізу текстових даних використано мову Matlab, яка містить різні вбудовані засоби для реалізації математичних обчислень та аналізу даних.

Розроблено структурну схему програми, відповідно до якої основними модулями програми є: головний модуль, модуль синтаксичного аналізу, модуль семантичного аналізу, модуль прагматичного аналізу, модуль редагування правил синтаксичного аналізу, модуль редагування правил семантичного аналізу, модуль редагування правил прагматичного аналізу, модуль побудови дерева аналізу.

Запропоновано функціональну схему програми, визначено, що програма дозволяє користувачу виконувати такі функції: синтаксичний аналіз; семантичний аналіз; прагматичний аналіз; побудова дерева аналізу; зміна правил побудови дерева синтаксичного аналізу; зміна правил побудови дерева семантичного аналізу; зміна правил побудови дерева прагматичного аналізу.

Наукова новизна роботи полягає у тому, що розроблено метод синтаксичного аналізу на основі діаграмного підходу, в якому після проведення аналізу кожного підрядка даних зберігаються результати, що

дозволяє не проводити повторний аналіз такого ж словосполучення в подальшому та скоротити час аналізу.

Практичне значення роботи полягає у тому, що розроблено програмне забезпечення для аналізу текстових даних. Розроблена програма дозволяє виконувати дослідження текстових даних шляхом синтаксичного, семантичного та прагматичного аналізу.

Виконано експериментальне дослідження методів аналізу текстових даних. За результатами порівняльного аналізу визначено, що запропонований метод працює швидше у порівнянні із висхідним методом та дозволяє отримувати релевантні результати аналізу текстових даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Анисимов А. Компьютерная лингвистика для всех: Язык.Алгоритмы. / А. Анисимов.– К.: Наук. думка, 2004.– 216 с.
2. Шемакин Ю.И. Начала компьютерной лингвистики /Ю.И.Шемакин.– М.: МГОУ, 2004.–343 с.
3. Исакадзе Н. В. Отражение морфологии и референциальной семантики именной группы в формальном синтаксисе /Н.В. Исакадзе.– М.: МГУ, 2005.–98 с.
4. Степанов М.Д. Русская компьютерная и квантитативная лингвистика /М.Д.Степанов.– М.:МГУ, 2004.– 516 с.
5. Шаров С.А. Средства компьютерного представления лингвистической информации /С.А. Шаров.– М: РНИИИИ, 2006.–765с.
6. Кривоносов А.Т. Язык, логика, мышление / А.Т.Кривоносов.– М., 2006.– 682 с.
7. Марчук Ю.Н. Вычислительная лексикография./ Ю.Н.Марчук.– М.: ВЦП, 1999, 175 с.
8. Пиотровский Р.Г. Инженерная лингвистика и теория языка. / Р.Г.Пиотровский– Л.: Наука, 2001.– 111 с.
9. Белоногов Г.Г. Автоматизация процессов накопления, поиска и обобщения информации. / Г.Г. Белоногов.– М.: Наука, 2003.– 255 с.
10. Половко А. М. Matlab для студента/ А. М. Половко, П. Н. Бутусов.– К.: Наукова думка, 2004. – 50 с.
11. Steinhaus S. Comparison of mathematical programs for data analysis / S. Steinhaus. – Munchen, Germany, 2004. – 65 p.
12. Смоленцев Н. К. Создание Windows-приложений с использованием математических процедур Matlab / Н. К. Смоленцев. – М.: ДМК-Пресс, 2008. – 456 с.
13. Потемкин В. Г. Система Matlab. Справочное пособие / В. Г. Потемкин. – М.: Диалог-МИФИ, 1997. – 350 с.

14. Экономика и организация производства программных продуктов [Электронный ресурс]. – 2019. – Режим доступа: <http://megaobzor.net/e-book/book.html> – (Экономика и организация производства программных продуктов)

15. Правила улаштування електроустановок [Текст]. – К. : Міненерговугілля України, 2017. – 617с.

16. Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом [Текст] : ДСТУ Б В.2.5-82:2016. – На заміну ДБН В.2.5-27-2006 ; чинний від 2017-04-01. – К. : ДП «УкрНДНЦ», 2016. –

109 с.

17. Голінько В.І. Основи охорони праці./ В.І. Голінько [Текст] – К. : Дніпропетровськ: Національний гірничий університет, 2010. – 271с.

18. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Електронний ресурс] : ДСанПіН 3.3.2.007-98. – Чинний від 1998-12-10. – К. : МОЗ України, 1998. – Режим доступу: <http://mozdocs.kiev.ua/view.php?id=2445>. – (Державні санітарні правила та норми)

19. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень. – Харків: Форт, 2000 – 24 с.

20. ДБН В.2.5-28:2018. Природне та штучне освітлення. Посібник: проблеми природного і штучного освітлення. – К: Етіп, 2019. – 180 с.

21. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку. – К. : МОЗ України, 1999 – 24 с.

22. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Електронний ресурс] : ДСанПіН 3.3.2.007-98. – Чинний від 1998-12-10. – К. : МОЗ України, 1998. – Режим доступу: <http://mozdocs.kiev.ua/view.php?id=2445>. – (Державні санітарні правила та норми)

23. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою [Текст] : ДСТУ Б В.1.1-36:2016. – На заміну НАПБ Б.03.002-2007 ; чинний від 2017-01-01. – К. : Мінрегіонбуд України, 2016. – 66 с. – (Державний Стандарт України)

24. Пожежна безпека об'єктів будівництва. Загальні вимоги [Текст] : ДБН В.1.1-7:2016. – На заміну ДБН В.1.1.7-2002 ; чинний від 2017-06-01. – К. : Мінрегіон України, 2017. – 47 с. – (Державні будівельні норми).

25. Жидецький В. Ц. Основи охорони праці [Текст] : підручник / В. Ц. Жидецький. – 5-те вид., доп. – К. : Знання, 2014. – 373 с. + 1 ел. опт. диск (CD-ROM). – ISBN 978-617-07-0134-3

26. Шоботов В. М. Цивільна оборона [Текст] : навчальний посібник / В. М. Шоботов. – Вид. 2-ге, перероб. – К. : Центр навчальної літератури, 2006. – 438 с.

ДОДАТОК А
Технічне завдання

Вступ

Програма може використовуватися для обробки лінгвістичної інформації.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну роботу на тему «Дослідження та програмна реалізація методів аналізу текстових даних», затверджене наказом по Національному університету «Запорізька політехніка».

A.2 Призначення розробки

Програма призначена для аналізу текстових даних шляхом синтаксичного, семантичного та прагматичного аналізу.

A.3 Вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма має виконувати такі функції:

- читання текстових даних з файлу;
- виконання синтаксичного аналізу;
- виконання семантичний аналізу;
- виконання прагматичний аналізу;
- побудова дерева аналізу;
- редагування правил побудови дерева синтаксичного аналізу;
- редагування правил побудови дерева семантичного аналізу;
- редагування правил побудови дерева прагматичного аналізу.

Інтерфейс користувача повинен забезпечити легке орієнтування у навігаційному меню, а також легке отримання вихідних даних.

А.3.2 Вимоги до надійності

Програма повинна коректно працювати, перевіряти на коректність тип вхідних даних. У разі невідповідності введених вхідних даних, видавати повідомлення про це.

А.3.3 Вимоги до експлуатації

Програма може бути записана як на твердому магнітному диску, так і на флеш носії. Експлуатація програмного продукту здійснюється відповідно до експлуатаційної документації на розроблену програму, що містить інформацію, необхідну для освоєння та експлуатації програмного продукту.

А.3.4 Вимоги до складу й параметрів технічних засобів

Для роботи з програмним продуктом рекомендується використовувати наступне апаратне забезпечення:

- ПК під керівництвом операційної системи;
- тактова частота процесора 1 ГГц або більше;
- обсяг оперативної пам'яті не менше 2 Гб;
- обсяг вільного простору на диску – не менше 2000 Мб;
- дозвіл екрана монітора не менше 1024x768;
- маніпулятор миша.

А.3.5 Вимоги до інформаційної й програмної сумісності

Для роботи програми необхідна операційна система Windows.

A.3.6 Вимоги до транспортування і зберігання

Програма повинна зберігатися на твердому магнітному диску або на флеш носії і транспортуватися в спеціальних боксах.

A.3.7 Вимоги до програмної документації

Програма повинна поставлятися з технічним завданням, керівництвом оператора, керівництвом програміста.

A.4 Стадії та етапи розробки

Програмний продукт повинен бути виконаний у 16-ти тижневий строк з моменту отримання завдання.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Програма розроблена за допомогою мови програмування Matlab.

Б.2 Функціональне призначення

Програма призначена для аналізу текстових даних шляхом синтаксичного, семантичного та прагматичного аналізу.

Б.3 Опис логічної структури

Програму розроблено на основі модульної структури. У проєкті використовуються такі модулі: головний модуль, модуль синтаксичного аналізу, модуль семантичного аналізу, модуль прагматичного аналізу, модуль редагування правил синтаксичного аналізу, модуль редагування правил семантичного аналізу, модуль редагування правил прагматичного аналізу, модуль побудови дерева аналізу.

Б.4 Використані технічні засоби

Для роботи з програмою рекомендується використовувати наступне апаратне забезпечення:

- ПК під керівництвом операційної системи;
- тактова частота процесора 1 ГГц або більше;
- обсяг оперативної пам'яті не менше 2 Гб;
- обсяг вільного простору на диску – не менше 2000 Мб;
- дозвіл екрана монітора не менше 1024x768;
- маніпулятор миша.

Б.5 Виклик і завантаження

Для звернення до програми необхідно відкрити з операційної системи файл MSL.fig, внаслідок чого буде запущено середовище MATLAB і відкритий графічний інтерфейс. Іншим способом є виконання програми з консолі по команді run ('MSL.m'), але для цього заздалегідь необхідно змінити поточну директорію на ту, в якій зберігається програма, або виклик з консолі окремих модулів та працювати з ними.

Б.6 Вхідні дані

Вхідними даними для роботи системи виступають: текст, який треба проаналізувати, набір правил формальної системи. Усі вказані вхідні дані знаходяться в окремих файлах, а в програмі вказуються шляхи до файлів, в яких зберігаються ці вхідні дані.

Б.7 Вихідні дані

Вихідними даними роботи програми є синтаксичні, семантичні, а також прагматичні структури проаналізованого тексту, які відображаються графічно.

ДОДАТОК В
Текст програми

```

function varargout = MSL(varargin)
% MSL M-file for MSL.fig
%     MSL, by itself, creates a new MSL or raises the existing
%     singleton*.
%
%     H = MSL returns the handle to a new MSL or the handle to
%     the existing singleton*.
%
%     MSL('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in MSL.M with the given input
arguments.
%
%     MSL('Property','Value',...) creates a new MSL or raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before MSL_OpeningFunction gets called. An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to MSL_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows
only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MSL

% Last Modified by GUIDE v2.5 24-Jan-2009 14:28:18

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @MSL_OpeningFcn, ...
                  'gui_OutputFcn',  @MSL_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before MSL is made visible.
function MSL_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MSL (see VARARGIN)

% Choose default command line output for MSL
handles.output = hObject;

```

```

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MSL wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = MSL_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
%In this function we load data from textfile and split them into
sentences and words.
%Then we save our data to file Syntax.mat
set(handles.text1,'String','Selecting file for syntax analysis');
isdir=1;
pat='.';
%matlabroot;
while (isdir>0)
    d=dir(pat);
    str = {d.name};
    [s,v] = listdlg('PromptString','Select a file:',...
                    'SelectionMode','single',...
                    'ListString',str);
    if (v==0)
        set(handles.text1,'String','File selection was canceled');
        return;
    end;
    isdir=d(s).isdir;
    pat=strcat(pat,'\ ',str(s));
    pat=pat{1};
end;
fid=fopen(pat,'r');
Sentences=[];
while 1
    tline=fgetl(fid);
    if ~ischar(tline)
        break;
    end;
    while ~isempty(tline)
        pos=find(tline==' ',1);
        if isempty(pos)
            pos=length(tline);
        end;
        Sentences{end+1}=tline(1:pos);
        tline(1:pos)=[];
    end;
    %disp(tline);
end

```

```

fclose(fid);
set(handles.listbox1,'String',Sentences);
set(handles.text1,'String','File for syntax analysis has been
loaded');

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% In this function we analis selected sentence with formal system e0

and
% draw tree for it
set(handles.text1,'String','Starting syntax analysis');
Sentences=get(handles.listbox1,'String');
if ~isempty(Sentences)
    Val=get(handles.listbox1,'Value');
    CurSent=Sentences{Val};
    CurSent(find(CurSent=='.'))=' ';
    CurSent(find(CurSent==','))=' ';
    Words={};
    while (CurSent(1)==' ')
        CurSent(1)=[];
    end;
    %     if CurSent(1)<char(91)
    %         CurSent(1)=char(CurSent(1)+32);
    %     end;
    while ~isempty(CurSent)
        while (CurSent(1)==' ')
            CurSent(1)=[];
        end;
        if isempty(CurSent)
            break;
        end;
        pos=find(CurSent==' ',1);
        if isempty(pos)
            Words{end+1}=CurSent;
        else
            Words{end+1}=CurSent(1:pos-1);
        end;
        CurSent(1:pos)=[];
    end;
    load('e0.mat','e0');
    S=Sint(Words,e0);
    save('Sint.mat','S');
    set(handles.text1,'String','Syntax analysis was finished');
else
    set(handles.text1,'String','No String for syntax analysis');
end;

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)

```

```

% hObject      handle to pushbutton5 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
set(handles.text1,'String','Starting semantic analysis');
load('Sint.mat','S');
if isempty(who('S'))
    set(handles.text1,'String','No data from syntax analysis found');
    return;
end;
load('e1.mat','e1');
if isempty(who('S'))
    set(handles.text1,'String','No rules for semantic data');
    return;
end;
ST=Sem(S,e1);
save('Sem.mat','ST');
set(handles.text1,'String','Semantic analysis was finished');

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton6 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton7 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
set(handles.text1,'String','Starting pragmatic analysis');
load('Sint.mat','S');
if isempty(who('S'))
    set(handles.text1,'String','No data from syntax analysis found');
    return;
end;
load('e2.mat','e2');
if isempty(who('S'))
    set(handles.text1,'String','No rules for pragmatic data');
    return;
end;
ST=Pragm(S,e2);
save('Pragm.mat','ST');
set(handles.text1,'String','Pragmatic analysis was finished');

% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject      handle to listbox1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox1 contents as
cell array
%         contents{get(hObject,'Value')} returns selected item from
listbox1

% --- Executes during object creation, after setting all properties.

```

```

function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text1,'String','Starting drawing tree of syntax
analysis');
load('Sint.mat','S');
figure;
ShowTree(S,6);
set(handles.text1,'String','Tree of syntax analysis is drawn');

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Drawing tree of semantic analysis
set(handles.text1,'String','Starting drawing tree of semantic
analysis');
load('Sem.mat','ST');
figure;
ShowTree(ST,6);
set(handles.text1,'String','Tree of semantic analysis is drawn');

% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.text1,'String','Starting drawing tree of pragmatic
analysis');
load('Pragm.mat','ST');
figure;
ShowTree(ST,6);
set(handles.text1,'String','Tree of pragmatic analysis is drawn');

% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
E0Edt;

% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)

```

```
% hObject    handle to pushbutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
E1Edt;
```

```
% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
E2Edt;
```

```
function S=Sint(words,e0)

%SINT Summary of this function goes here
% Detailed explanation goes here
if (nargin==0)
    words={'the' 'wumpus' 'is' 'dead'};
    load('e00.mat');
end;

%S=struct('Name','S','Length',2,'Value',[struct('Name','NP','Length',2
,'Value',[struct('Name','Article','Length',1,'Value','the'),struct('Name','No
un','Length',1,'Value','wumpus')]),struct('Name','VP','Length',2,'Value',[str
uct('Name','Verb','Length',1,'Value','is'),struct('Name','Adjective','Length'
,1,'Value','dead')])]);

S=words;
k=1;
while (k>0)
    k=0;
    for i=1:length(S)
        for j=1:length(e0)
            L=length(e0(j).Value);
            if (i+L-1>length(S))
                continue;
            end;
            for z=1:L
                if isstruct(S{i+z-1})
                    T=strcmp(S{i+z-1}.Name,e0(j).Value(z));
                else
                    T=strcmp(S{i+z-1},e0(j).Value(z));
                end;
                k=k+T;
            end;
            if (k==L)
                SubS=S(i:(i+L-1));
                S{i}=struct('Name',e0(j).Name,'Length',L,'Value','');
                S{i}.Value=SubS;
            end;
        end;
    end;
end;
```

```

                S((i+1):(i+L-1))=[];
                break;
            else
                k=0;
            end;
        end;
    end;
    if (k>0)
        break;
    end;
end;

end;

SubS=S;
S{1}=struct('Name','S','Length',length(S),'Value','');
S{1}.Value=SubS;
S(2:end)=[];

function S=Sem(SNT,e1)
%SINT Summary of this function goes here
% Detailed explanation goes here
if (nargin==0)
    load('Sint.mat','S');
    SNT=S;
    load('e1.mat','e1');
end;
T=SNT;
while (1)
    i=1;
    j=1;
    while (1)
        K=T{j};
        if (i(j)>K.Length)
            j=j-1;
            if (j==0)
                break;
            else
                T{j}.Value{i(j)}=T{j+1};
                i(j)=i(j)+1;
                continue;
            end;
        end;
    end;
    if isstruct(K.Value{i(j)})
        Sema=0;
        Q=0;
        for zzz=1:K.Length
            if isfield(K.Value{zzz},'Name2')
                Q=Q+1;
                if ~isempty(find(K.Value{zzz}.Name2=='('))
                    Sema=zzz;
                end;
            end;
        end;
        if (Q==K.Length)
            if (Sema>0)
                Str=K.Value{Sema}.Name2;
                SubStr=Str(1:(find(Str==' ',1)-1));
            end;
        end;
    end;
end;

```

```

        for zzz=1:K.Length
            if
                (zzz~=Sema) && (strcmp(K.Value{zzz}.Name, SubStr))
                    Str(1:(length(SubStr)+1))=[];
                    pos=strfind(Str, SubStr);
                    if ~isempty(pos)
                        pos=pos(end);
                    end;
                    Str=char(strcat(Str(1:pos-
1), K.Value{zzz}.Name2, Str((pos+length(SubStr)):end)));
                    end;
                end;
                T{j}.Name2=Str;
            else
                T{j}.Name2=K.Value{1}.Name2;
            end;
        end;
        T{j+1}=K.Value{i(j)};
        j=j+1;
        i(j)=1;
        continue;
    else
        Cons=[];
        for q=1:length(e1)
            if strcmp(e1(q).Text, K.Value)
                Cons=e1(q);
                break;
            end;
        end;
        if ~isempty(Cons)
            tmp=Cons.Params;
            tmp(find(tmp==' '))=',';
            T{j}.Name2=char([Cons.Params ' *' Cons.Text '(' tmp
')']);

            else
                T{j}.Name2=T{j}.Value{1};
            end;
        end;
        i(j)=i(j)+1;
    end;
    if isfield(T{1}, 'Name2')
        break;
    end;
end;
i=1;
j=1;
T={T{1}};
while (1)
    K=T{j};
    if (i(j)>K.Length)
        j=j-1;
        if (j==0)
            break;
        else
            T{j}.Value{i(j)}=T{j+1};
            i(j)=i(j)+1;
            continue;
        end;
    end;
    if isstruct(K.Value{i(j)})
        if (isfield(T{j}, 'Name2')) && (isempty(find(T{j}.Name=='(')))
            T{j}.Name=[T{j}.Name ' (' T{j}.Name2 ')'];
        end;
        T{j+1}=K.Value{i(j)};
    end;
end;

```

```

        j=j+1;
        i(j)=1;
        continue;
    else
        %disp(K.Value);
        if isfield(T{j}, 'Name2')
            T{j}.Name=[T{j}.Name ' (' T{j}.Name2 ')'];
        end;
    end;
    i(j)=i(j)+1;
end;
S={T{1}};

```

```

function S= Pragm(SNT,e2)
% Pragm Summary of this function goes here
% Detailed explanation goes here
if (nargin==0)
    load('Sint.mat','S');
    SNT=S;
    load('e2.mat','e2');
end;
i=1;
j=1;
T=SNT;
while (1)
    K=T{j};
    if (i(j)>K.Length)
        j=j-1;
        if (j==0)
            break;
        else
            T{j}.Value{i(j)}=T{j+1};
            i(j)=i(j)+1;
            continue;
        end;
    end;
    if isstruct(K.Value{i(j)})
        T{j+1}=K.Value{i(j)};
        j=j+1;
        i(j)=1;
        continue;
    else
        Cons=[];
        for q=1:length(e2)
            if strcmp(e2(q).Text,K.Value)
                Cons=e2(q);
                break;
            end;
        end;
    end;
end;

```

```

        end;
    end;
    if ~isempty(Cons)
        T{j}.Name3=char(Cons.Value);
    end;
end;
i(j)=i(j)+1;
end;

i=1;
j=1;
T={T{1}};
while (1)
    K=T{j};
    if (i(j)>K.Length)
        j=j-1;
        if (j==0)
            break;
        else
            T{j}.Value{i(j)}=T{j+1};
            i(j)=i(j)+1;
            continue;
        end;
    end;
    if isstruct(K.Value{i(j)})
        if (isfield(T{j}, 'Name3')) && (isempty(find(T{j}.Name=='(')))
            T{j}.Name=[T{j}.Name ' (' T{j}.Name3 ')'];
        end;
        T{j+1}=K.Value{i(j)};
        j=j+1;
        i(j)=1;
        continue;
    else
        %disp(K.Value);
        if isfield(T{j}, 'Name3')
            T{j}.Value=[T{j}.Value ' (' T{j}.Name3 ')'];
        end;
    end;
    i(j)=i(j)+1;
end;
S={T{1}};

```

```

function ShowTree(Tree,levels)
axes;
set(gca,'XTick',[],'YTick',[]);
if (nargin==0)

%Tree={'S',{'NP',{'Article','the'},{'Noun','wumpus'}},{ 'VP',{'Verb','is'},{'A
djective','dead'}}};

Tree=struct('Name','S','Length',2,'Value',[struct('Name','NP','Length',2,'Val
ue',[struct('Name','Article','Length',1,'Value','the'),struct('Name','Noun','
Length',1,'Value','wumpus')]),struct('Name','VP','Length',2,'Value',[struct('
Name','Verb','Length',1,'Value','is'),struct('Name','Adjective','Length',1,'V
alue','dead')])]);

    levels=3;
end;
i=1;
j=1;
T=Tree;
y0=1;
dy=y0/levels;
Parx=struct('minx',0,'maxx',1);
text(0.5,0.98,T{1}.Name,'HorizontalAlignment','center');
while (1)
    K=T{j};
    Nm=K.Name;
    if isempty(Nm)
        K=Parx(j);
        K.minx=0;
        K.maxx=1;
        K=T{j};
    end;
    if (i(j)>K.Length)
        j=j-1;
        if (j==0)
            break;
        else
            i(j)=i(j)+1;
            continue;
        end;
    end;
    if isstruct(K.Value{i(j)})
        T{j+1}=K.Value{i(j)};
        j=j+1;
    end;
end;

```

```

        i(j)=1;
        minx=Parx(j-1).minx+(Parx(j-1).maxx-Parx(j-1).minx)/T{j-1}.Length*(i(j-1)-1);
        maxx=Parx(j-1).minx+(Parx(j-1).maxx-Parx(j-1).minx)/T{j-1}.Length*(i(j-1));
        Parx(j)=struct('minx',minx,'maxx',maxx);
        y=y0-dy*(j-1);
        text((minx+maxx)/2,y,T{j}.Name,'HorizontalAlignment','center');
        line([(Parx(j-1).minx+Parx(j-1).maxx)/2
(Parx(j).minx+Parx(j).maxx)/2],[y0-dy*(j-2) y0-dy*(j-1)]);
        continue;
    else

text((Parx(j).maxx+Parx(j).minx)/2,0.05,K.Value,'color','red','HorizontalAlignment','center');

        line([(Parx(j).minx+Parx(j).maxx)/2
(Parx(j).minx+Parx(j).maxx)/2],[y0-dy*(j-1) 0.06]);
        %disp(K.Value);
    end;
    i(j)=i(j)+1;
end;

```

Д.5 Текст модуля E0Edt.m

```

function varargout = E0Edt(varargin)
% E0EDT M-file for E0Edt.fig
%   E0EDT, by itself, creates a new E0EDT or raises the existing
%   singleton*.
%
%   H = E0EDT returns the handle to a new E0EDT or the handle to
%   the existing singleton*.
%
%   E0EDT('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in E0EDT.M with the given input
arguments.
%
%   E0EDT('Property','Value',...) creates a new E0EDT or raises the
%   existing singleton*. Starting from the left, property value
pairs are
%   applied to the GUI before E0Edt_OpeningFunction gets called.
An
%   unrecognized property name or invalid value makes property
application
%   stop. All inputs are passed to E0Edt_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows
only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

```

```

% Edit the above text to modify the response to help E0Edt

% Last Modified by GUIDE v2.5 25-Jan-2009 09:53:43

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @E0Edt_OpeningFcn, ...
                  'gui_OutputFcn',  @E0Edt_OutputFcn, ...
                  'gui_LayoutFcn',  [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before E0Edt is made visible.
function E0Edt_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin    command line arguments to E0Edt (see VARARGIN)

% Choose default command line output for E0Edt
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
load('e0.mat','e0');
if isempty(who('e0'))
    e0=struct();
end;
Names=cell(length(e0),1);
Values=cell(length(e0),1);
for i=1:length(e0)
    Names{i}=e0(i).Name;
    Values{i}=e0(i).Value{1};
    for j=2:length(e0(i).Value)
        Values{i}=char([Values{i}, ' ',e0(i).Value{j}]);
    end;
end;
set(handles.listbox1,'String',Values);
S=[handles.radiobutton2 handles.radiobutton3 handles.radiobutton4 ...
    handles.radiobutton5 handles.radiobutton6 handles.radiobutton7 ...
    handles.radiobutton8 handles.radiobutton9 handles.radiobutton10
...
    handles.radiobutton11 handles.radiobutton12 handles.radiobutton13
handles.radiobutton15];
handles.S=S;
handles.e0=e0;
guidata(hObject,handles);

```

```

%set(handles.listbox2,'String',Names);
% UIWAIT makes E0Edt wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = E0Edt_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject handle to listbox1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
j=get(hObject,'Value');
for i=1:length(handles.S)
    if strcmp(handles.e0(j).Name,get(handles.S(i),'String'))
        set(handles.S(i),'Value',1);
    else
        set(handles.S(i),'Value',0);
    end;
end;
% Hints: contents = get(hObject,'String') returns listbox1 contents as
cell array
% contents{get(hObject,'Value')} returns selected item from
listbox1

% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject handle to listbox1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
answer=inputdlg('Введите список правил, разделенных пробелом:
','Список правил');
words={};
answer=answer{1};

```

```

set(handles.listbox1,'String',[get(handles.listbox1,'String');answer])
;

while ~isempty(answer)
    while (answer(1)==' ')
        answer(1)=[];
    end;
    p=find(answer==' ',1);
    if isempty(p)
        words{end+1}=answer;
        answer='';
    else
        words{end+1}=answer(1:p-1);
        answer(1:p)=[];
    end;
end;
e0=handles.e0;
e0(end+1).Name='Noun';
e0(end).Value=words;
e0(end).Type=0;
handles.e0=e0;
guidata(hObject,handles);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
j=get(handles.listbox1,'Value');
if (j<1)
    return;
end;
Q=questdlg('Are you sure that you want to delete this
item?','Confirmation','Yes','No','No');
if ~strcmp(Q,'Yes')
    return;
end;
S=get(handles.listbox1,'String');
S(j)=[];
set(handles.listbox1,'String',S);
handles.e0(j)=[];
guidata(hObject,handles);

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
e0=handles.e0;
for i=1:length(e0)
    tmp=0;
    for j=1:length(handles.S)
        if strcmp(e0(i).Name,get(handles.S(j),'String'))
            tmp=1;
            break;
        end;
    end;
    if (tmp==0)
        warndlg(['Unknown type of ',e0(i).Value,'?']);
        return;
    end;
end;

```

```

end;
save('e0.mat','e0');
close(handles.figure1);

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton4 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
close(handles.figure1);

% --- Executes on button press in radiobutton2.
function radiobutton_Callback(hObject, eventdata, handles)
% hObject      handle to radiobutton2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
j=get(handles.listbox1,'Value');
handles.e0(j).Name=get(hObject,'String');
guidata(hObject,handles);

function varargout = E1Edt(varargin)
% E1EDT M-file for E2Edt.fig
%     E1EDT, by itself, creates a new E1EDT or raises the existing
%     singleton*.
%
%     H = E1EDT returns the handle to a new E1EDT or the handle to
%     the existing singleton*.
%
%     E1EDT('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in E1EDT.M with the given input
arguments.
%
%     E1EDT('Property','Value',...) creates a new E1EDT or raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before E1Edt_OpeningFunction gets called.
An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to E1Edt_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows
only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help E1Edt

% Last Modified by GUIDE v2.5 26-Jan-2009 00:06:42

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @E1Edt_OpeningFcn, ...
                  'gui_OutputFcn',  @E1Edt_OutputFcn, ...
                  'gui_LayoutFcn',  [] , ...

```

```

        'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before ElEdt is made visible.
function ElEdt_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ElEdt (see VARARGIN)

% Choose default command line output for ElEdt
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
load('e1.mat','e1');
if isempty(who('e1'))
    e1=struct();
end;
Text=cell(length(e1),1);
Params=cell(length(e1),1);
for i=1:length(e1)
    Text{i}=e1(i).Text;
    Params{i}=e1(i).Params;
end;
set(handles.listbox1,'String',Text);
set(handles.edit2,'String',Params{1});
handles.e1=e1;
guidata(hObject,handles);
%set(handles.listbox2,'String',Names);
% UIWAIT makes ElEdt wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = ElEdt_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

```

```

% handles      structure with handles and user data (see GUIDATA)
j=get(hObject,'Value');
set(handles.edit2,'String',handles.e1(j).Params);
% Hints: contents = get(hObject,'String') returns listbox1 contents as
cell array
%           contents{get(hObject,'Value')} returns selected item from
listbox1

% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject      handle to listbox1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%           See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton1 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
answer=inputdlg('Введите прагмат: ','Прагмат');
answer=answer{1};
set(handles.listbox1,'String',[get(handles.listbox1,'String');answer])
;

e1=handles.e1;
e1(end+1).Text=answer;
e1(end).Params='';
handles.e1=e1;
guidata(hObject,handles);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
j=get(handles.listbox1,'Value');
if (j<1)
    return;
end;
Q=questdlg('Are you sure that you want to delete this
item?', 'Confirmation','Yes','No','No');
if ~strcmp(Q,'Yes')
    return;
end;
S=get(handles.listbox1,'String');
S(j)=[];
set(handles.listbox1,'String',S);
handles.E1(j)=[];
guidata(hObject,handles);

```

```

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton3 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
e1=handles.e1;
for i=1:length(e1)
    if isempty(e1(i).Params)
        warndlg(['Неизвестное значение для прагмата ' e1(i).Params]);
        return;
    end;
end;
save('e1.mat','e1');
close(handles.figure1);

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton4 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
close(handles.figure1);

function edit2_Callback(hObject, eventdata, handles)
% hObject      handle to edit2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject      handle to edit2 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton6 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
LB=get(handles.listbox1,'Value');
if (LB<=0)
    retrun;
end;

```

```

handles.e1(LB).Params=get(handles.edit2,'String');
guidata(hObject,handles);

function varargout = E2Edt(varargin)
% E2EDT M-file for E2Edt.fig
%     E2EDT, by itself, creates a new E2EDT or raises the existing
%     singleton*.
%
%     H = E2EDT returns the handle to a new E2EDT or the handle to
%     the existing singleton*.
%
%     E2EDT('CALLBACK',hObject,eventData,handles,...) calls the local
%     function named CALLBACK in E2EDT.M with the given input
arguments.
%
%     E2EDT('Property','Value',...) creates a new E2EDT or raises the
%     existing singleton*. Starting from the left, property value
pairs are
%     applied to the GUI before E2Edt_OpeningFunction gets called.
An
%     unrecognized property name or invalid value makes property
application
%     stop. All inputs are passed to E2Edt_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows
only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help E2Edt

% Last Modified by GUIDE v2.5 26-Jan-2009 00:06:42

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn',  @E2Edt_OpeningFcn, ...
                  'gui_OutputFcn',  @E2Edt_OutputFcn, ...
                  'gui_LayoutFcn',  [], ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before E2Edt is made visible.
function E2Edt_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```

% varargin    command line arguments to E2Edt (see VARARGIN)

% Choose default command line output for E2Edt
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
load('e2.mat','e2');
if isempty(who('e2'))
    e2=struct();
end;
Text=cell(length(e2),1);
Values=cell(length(e2),1);
for i=1:length(e2)
    Text{i}=e2(i).Text;
    Values{i}=e2(i).Value;
end;
set(handles.listbox1,'String',Text);
set(handles.edit2,'String',Values{1});
handles.e2=e2;
guidata(hObject,handles);
%set(handles.listbox2,'String',Names);
% UIWAIT makes E2Edt wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = E2Edt_OutputFcn(hObject, eventdata, handles)
% varargout    cell array for returning output args (see VARARGOUT);
% hObject     handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject     handle to listbox1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
j=get(hObject,'Value');
set(handles.edit2,'String',handles.e2(j).Value);
% Hints: contents = get(hObject,'String') returns listbox1 contents as
cell array
%           contents{get(hObject,'Value')} returns selected item from
listbox1

% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject     handle to listbox1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.

```

```

        if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
            set(hObject,'BackgroundColor','white');
        end

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
answer=inputdlg('Введите прармат: ','Прармат');
answer=answer{1};
set(handles.listbox1,'String',[get(handles.listbox1,'String');answer])
;

e2=handles.e2;
e2(end+1).Text=answer;
e2(end).Value='';
handles.e2=e2;
guidata(hObject,handles);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
j=get(handles.listbox1,'Value');
if (j<1)
    return;
end;
Q=questdlg('Are you sure that you want to delete this
item?','Confirmation','Yes','No','No');
if ~strcmp(Q,'Yes')
    return;
end;
S=get(handles.listbox1,'String');
S(j)=[];
set(handles.listbox1,'String',S);
handles.e2(j)=[];
guidata(hObject,handles);

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
e2=handles.e2;
for i=1:length(e2)
    if isempty(e2(i).Value)
        warndlg(['Неизвестное значение для прармата ' e2(i).Value]);
        return;
    end;
end;
save('e2.mat','e2');
close(handles.figure1);

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
close(handles.figure1);

function edit2_Callback(hObject, eventdata, handles)
% hObject handle to edit2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
% str2double(get(hObject,'String')) returns contents of edit2
as a double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject handle to edit2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject handle to pushbutton6 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
LB=get(handles.listbox1,'Value');
if (LB<=0)
retrun;
end;
handles.e2(LB).Value=get(handles.edit2,'String');
guidata(hObject,handles);

```

ДОДАТОК Г
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дипломна кваліфікаційна робота

**Дослідження та програмна реалізація методів аналізу
текстових даних**

Виконав
ст. гр. КНТ-119м

А. П. Дєдик

Керівник
к.т.н., доцент

А. О. Олійник

Рисунок Г.1 – Слайд № 1

Об'єктом роботи є процес обробки текстових даних.

Предмет дослідження – методи аналізу текстових даних.

Мета роботи – дослідження та розробка засобів аналізу текстових даних.

Рисунок Г.2 – Слайд № 2

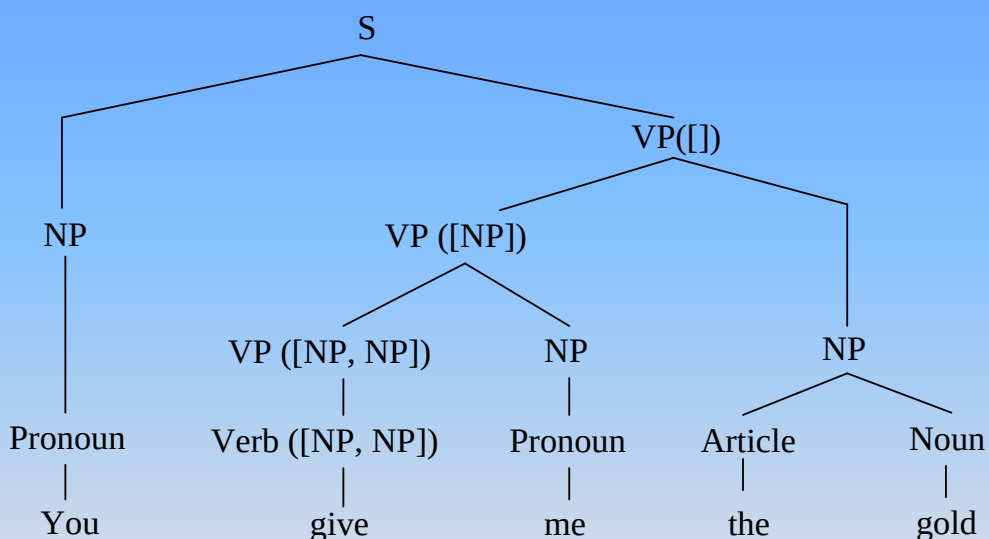
Задачі роботи:

- дослідження методів аналізу текстових даних;
- розробка технічного завдання для реалізації програми;
- вибір середовища розробки програмного забезпечення;
- розробка методу синтаксичного аналізу на основі діаграмного підходу;
- розробка структурної схеми програми;
- конструювання програмного забезпечення для аналізу текстових даних;
- тестування розробленого програмного забезпечення.

3

Рисунок Г.3 – Слайд № 3

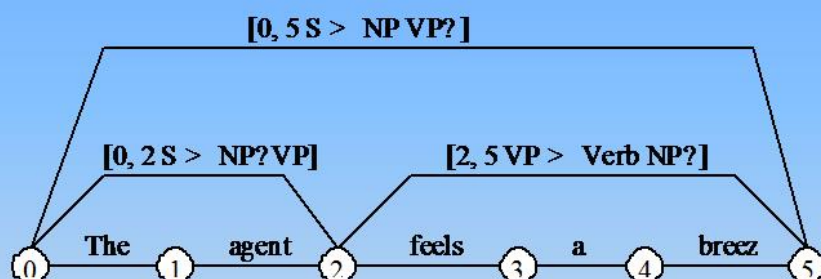
Дерево синтаксичного аналізу речення



4

Рисунок Г.4 – Слайд № 4

Метод синтаксичного аналізу на основі діаграмного підходу



5

Рисунок Г.5 – Слайд № 5

Вибір мови програмування [1]

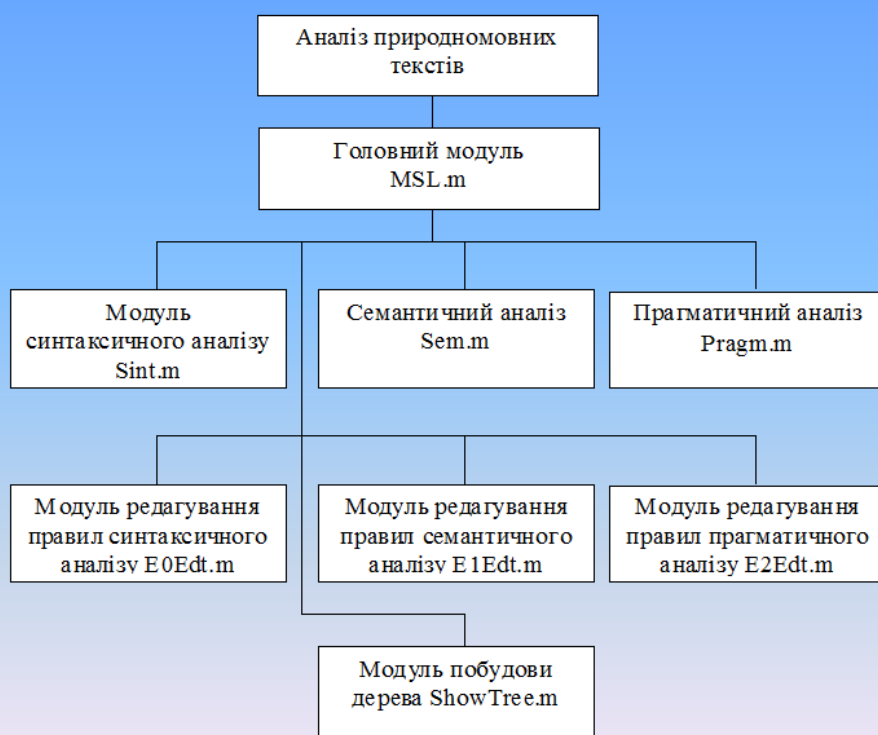
Критерій порівняння	MATLAB	SciLab	MuPAD
Ліцензія	Пропрієтарна	GPL	Пропрієтарна
Математичні функції	69,15%	39,74%	51,04%
Стандартна математика	100,00%	80,00%	100,00%
Алгебра	95,29%	76,47%	83,24%
Аналіз	100,00%	100,00%	100,00%
Чисельна математика	80,00%	57,14%	85,71%
Статистика	52,64%	19,09%	20,09%
Інші	61,60%	4,00%	7,60%
Робота з графіками	87,19%	48,74%	42,96%
Типи графіків	85,29%	41,18%	38,24%
Експорт/імпорт графіків	92,86%	71,43%	57,14%
Підтримка OLE	+	-	+
C/C++	+	+	+
DLL-виклики	+	+	-

1. Steinhaus S. Comparison of mathematical programs for data analysis /
S. Steinhaus. – Munchen, Germany. – 65 p.

6

Рисунок Г.6 – Слайд № 6

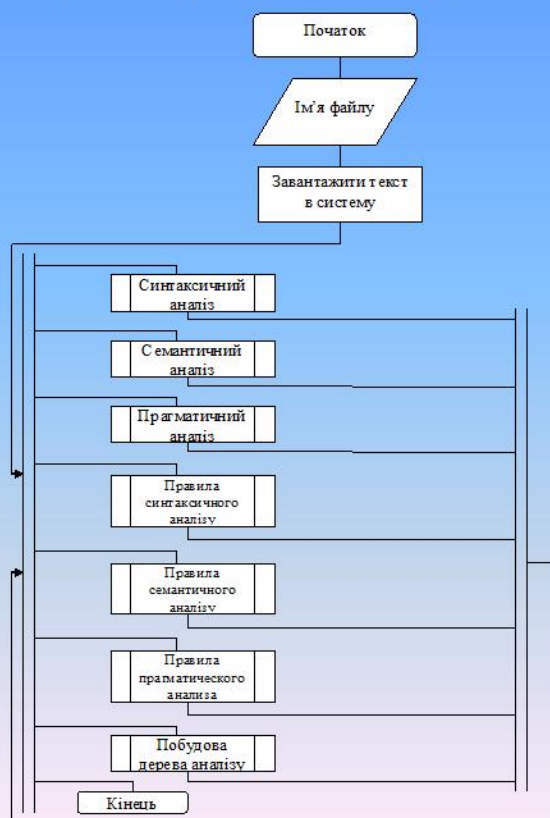
Структурна схема програми



7

Рисунок Г.7 – Слайд № 7

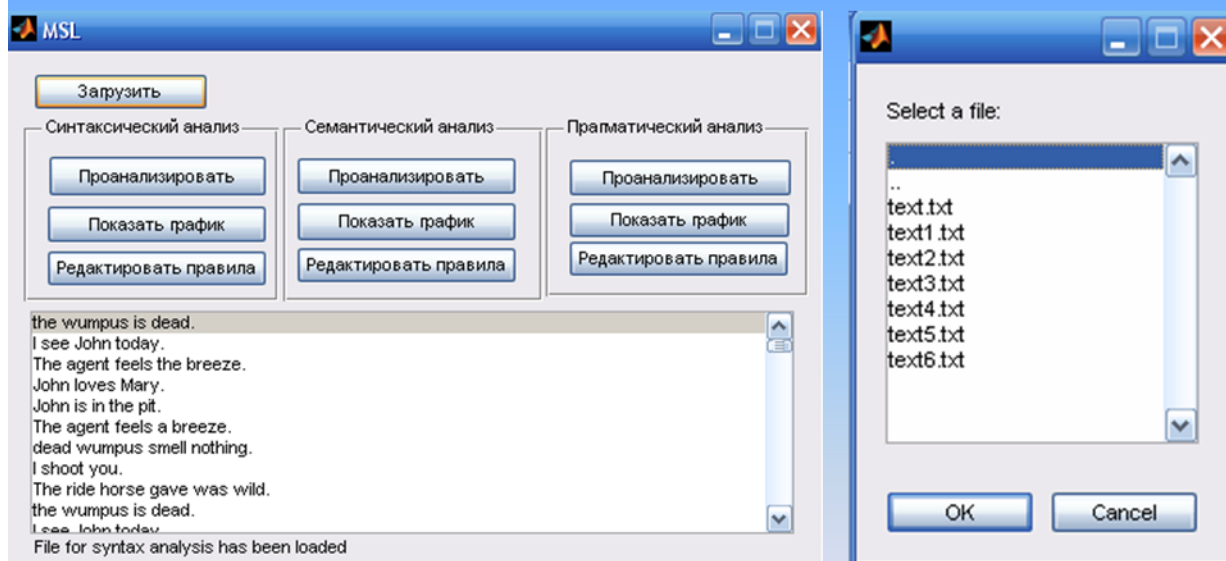
Схема алгоритму функціонування програми



8

Рисунок Г.8 – Слайд № 8

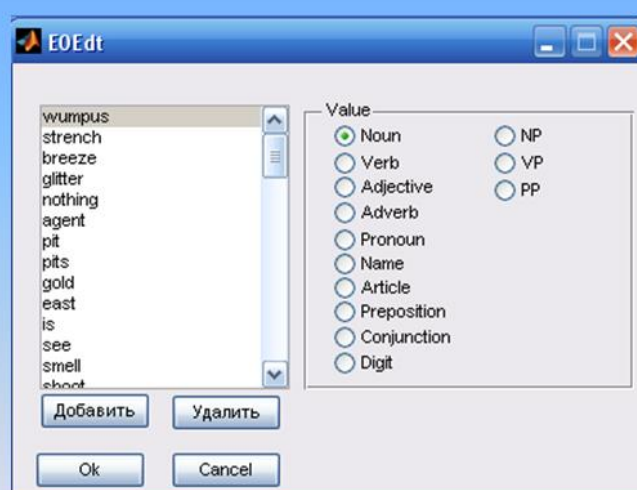
Робота з програмою. Головне вікно та вікно вибору файлу



9

Рисунок Г.9 – Слайд № 9

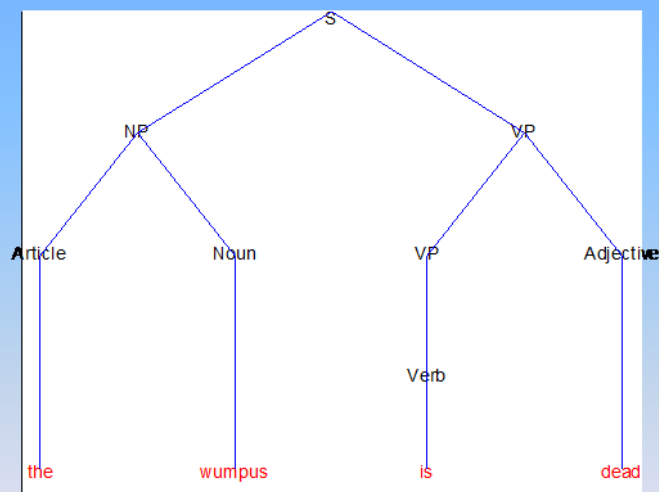
Робота з програмою. Редагування правил синтаксичного розбору



10

Рисунок Г.10 – Слайд № 10

Експерименти та результати. Побудоване дерево синтаксичного розбору



11

Рисунок Г.11 – Слайд № 11

Експерименти та результати. Порівняння методів аналізу текстових даних

Метод / Критерій	Висхідний метод	Спадний метод	Метод синтаксичного аналізу на основі діаграмного підходу
Час роботи, с	25,776 с	21,776 с	22,776 с
Рівень точності побудови	Висока. Більшість схем релевантні	Середня. Іноді зустрічаються нерелевантні схеми	Висока. Більшість схем релевантні

12

Рисунок Г.12 – Слайд № 12

Висновки

У дипломній кваліфікаційній роботі магістра досліджено та програмно реалізовано методи аналізу текстових даних.

Розв'язано такі завдання:

- дослідження методів аналізу текстових даних;
- розробка технічного завдання для реалізації програми;
- вибір середовища розробки програмного забезпечення;
- розробка методу синтаксичного аналізу на основі діаграмного підходу;
- розробка структурної схеми програми;
- конструювання програмного забезпечення для аналізу текстових даних;
- тестування розробленого програмного забезпечення.

13

Рисунок Г.13 – Слайд № 13

Висновки

Проаналізовано процес обробки текстових даних. Досліджено формальні граматики. Визначено, що формальною граматикою називають спосіб опису формальної мови, тобто виділення деякої підмножини з множини всіх можливих слів деякого кінцевого алфавіту. Проаналізовано методи синтаксичного аналізу, розширені граматики, семантичну інтерпретацію.

Наукова новизна роботи полягає у тому, що розроблено метод синтаксичного аналізу на основі діаграмного підходу, в якому після проведення аналізу кожного підрядка даних зберігаються результати, що дозволяє не проводити повторний аналіз такого ж словосполучення в подальшому та скоротити час аналізу.

Практичне значення роботи полягає у тому, що розроблено програмне забезпечення для аналізу текстових даних. Розроблена програма дозволяє виконувати дослідження текстових даних шляхом синтаксичного, семантичного та прагматичного аналізу.

Результати тестування розробленої програми підтвердили її працездатність та відповідність вимогам технічного завдання.

14

Рисунок Г.14 – Слайд № 14