Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи дистанційного голосового керування робототехнічним комплексом”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Гуровський В.Д.
«____» _____ 2023 р.

Керівник проекту
доктор технічних наук, доцент
_____ Коваленко О.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Гуровському Віталію Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи дистанційного голосового керування робототехнічним комплексом*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 34-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи дистанційного голосового керування робототехнічним комплексом*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Гуровський В.Д. Дослідження та програмна реалізація системи дистанційного голосового керування робототехнічним комплексом. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи дистанційного голосового керування робототехнічним комплексом.

Метою розробки є дослідження та програмна реалізація системи дистанційного голосового керування робототехнічним комплексом.

Об'єктом дослідження є процес дистанційного голосового керування робототехнічним комплексом.

Предметом дослідження є методи дистанційного голосового керування робототехнічним комплексом.

Методи дослідження базуються на методах теорії Інтернету речей, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи дистанційного голосового керування робототехнічним комплексом.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Visual C++.

Ключові слова: комп'ютерна інженерія, керування, робототехнічний комплекс

ABSTRACT

Hurovskyi V.D. Research and software implementation of a system of remote voice control of a robotic complex. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software was developed, which is intended for the remote voice control system of a robotics complex.

The purpose of the development is the research and software implementation of the system of remote voice control of the robotic complex.

The object of research is the process of remote voice control of a robotic complex.

The subject of research is methods of remote voice control of a robotic complex.

Research methods are based on methods of the Internet of Things theory, methods of mathematical statistics, and methods of software development.

The result of the work is the software implementation of the system of remote voice control of the robotics complex.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Visual C++ environment.

Keywords: computer engineering, control, robotic complex

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЛП	–	лінійне пророкування
МС	–	мовний сигнал
ПЗ	–	програмне забезпечення
ПК	–	персональний комп’ютер
ПР	–	промисловий робот
РТД	–	роботизована технологічна дільниця
РТК	–	робототехнічний комплекс
РТЛ	–	роботизована технологічна лінію
СКВ	–	середнє квадратичне відхилення
СРМ	–	система розпізнавання мови

ВСТУП

Актуальність теми. У цей час у міру росту обсягів інформації комп'ютерна техніка усе більше й більше проникає в людське життя. Відбувається вдосконалювання інтерфейсу людина-комп'ютер. Винаходяться нові способи відображення інформації, модернізуються пристрої уведення, тривають пошуки такого інтерфейсу, що влаштував би всіх. На цю роль зараз претендує мовний інтерфейс. Власне кажучи, це саме те, до чого людство завжди прагнуло в спілкуванні з комп'ютером.

Роботи в цьому напрямку велися ще в той час, коли про графічний інтерфейс ніхто навіть і не думав. За порівняно короткий період був вироблений вичерпний теоретичний базис, і практичні досягнення обумовлювали тільки продуктивністю комп'ютерної техніки. В 60-70х роках були створені пристрої, здатні розпізнавати десяток мовних команд.

Сучасні розробки, як правило, ґрунтуються на біонічній моделі сприйняття мови людиною. Такі системи є ієрархічними, детермінованими, з навчанням і складаються з декількох взаємозалежних рівнів. Виділяються акустична (одержання первинних ознак мовних сигналів) і лінгвістична (робота зі словниками) складові.

Системи розпізнавання злитої мови будуються на базі імовірнісних моделей граматики мови. На словниках обсягом до 5000 слів вірогідність розпізнавання цілих фраз становить більше 95%, що вважається достатнім для забезпечення успішного мовного уведення тексту на ПК.

Для завдання голосового керування різними пристроями необхідно розпізнавання окремих мовних команд. Як правило, такий спосіб керування вимагає високої надійності (99% точності розпізнавання). Найчастіше команди вимовляються в умовах підвищеної зашумленість, наприклад на виробництві. Сучасні розробки в лабораторних умовах досягають 95% точності на словниках

до 100 команд і вимагають навчальні вибірки більших обсягів (10 і більше варіантів проголошення кожного слова різними дикторами).

Таким чином, проблема побудови ефективних алгоритмів розпізнавання мовних команд є актуальною.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи дистанційного голосового керування робототехнічним комплексом.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем дистанційного голосового керування робототехнічним комплексом.

– Дослідження системи дистанційного голосового керування робототехнічним комплексом.

– Програмна реалізація системи дистанційного голосового керування робототехнічним комплексом.

Об'єктом дослідження є процес дистанційного голосового керування робототехнічним комплексом.

Предметом дослідження є методи дистанційного голосового керування робототехнічним комплексом.

Методи дослідження базуються на методах теорії Інтернету речей, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод дистанційного голосового керування робототехнічним комплексом.

– Розроблено вітчизняний продукт дистанційного голосового керування робототехнічним комплексом, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі дистанційного голосового керування робототехнічним комплексом.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи дистанційного голосового керування робототехнічним комплексом, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Сучасні системи розпізнавання мови (СРМ), як правило, мають ієрархічну модульну структуру. На першому рівні виконується попередня обробка – виділення акустичних ознак, що характеризують мовні сигнали. Одним з найбільше часто використовуваних методів є лінійне пророкування (ЛП). Отримані на основі ЛП ознаки володіють рядом корисних властивостей – вони просто розраховуються, дають компактне подання мовного сигналу (МС), найменш чутливі до дій перешкод.

Наступний рівень СРМ є лінгвістичним. У нього входить процедура пошуку по словниках еталонів. У завданнях розпізнавання злитої мови будуються імовірнісні граматики мови, завдяки чому досягається високий ступінь розпізнавання цілих фраз.

При розпізнаванні окремих мовних команд слово вимовляється диктором без навколишнього контексту. Навчання таких систем є трудомістким процесом. Для підвищення надійності звичайно використовуються великі навчальні вибірки (10 і більше варіантів проголошення одного слова різними дикторами). Кожне слово моделюється схованою марківською моделлю або нейронною мережею.

При побудові систем, орієнтованих на одного диктора, можливе використання більше простого методу пошуку по словниках – нелінійного часового вирівнювання (динамічного програмування). У такому випадку в процесі навчання кожний еталон записується тільки один раз.

У магістерській роботі наведена класифікація мовних одиниць: фонеми й алофони, склади, цілі слова й фрази. Зроблено вивід, що мінімальну еталонну мовну одиницю варто вибирати залежно від призначення СРМ.

1.2 Область застосування

Областю застосування є робототехнічні комплекси (РТК). Найпростішим типом РТК є роботизована технологічна ланка (одиниця роботизованого устаткування), де виконується певна кількість допоміжних технологічних операцій.

Більш складним РТК є роботизована технологічна дільниця (РТД), яка об'єднує кілька роботизованих одиниць устаткування. На РТД промислові роботи виконують низку допоміжних технологічних операцій. Якщо операції здійснюються в єдиному технологічному процесі, то комплекс являє собою роботизовану технологічну лінію (РТЛ).

Сукупність РТД може являти собою цех, що охоплює також кілька автоматизованих складів і транспортних ПР, що зв'язує їх. Вищою формою розвитку роботизованого виробництва є комплексно роботизований завод.

Промислові роботи в РТК можуть виконувати основні технологічні операції (складання, зварювання, фарбування і т. д.) або допоміжні – з обслуговування основного технологічного устаткування. Серійність і номенклатура продукції визначаються розміром партії, що може випускатися без переналагодження комплексу, і переліком видів продукції, що випускаються. Кожний робототехнічний комплекс характеризується граничними значеннями цих параметрів. Розрізняють РТК із централізованим, децентралізованим і комбінованим управлінням. Людина в РТК може безпосередньо брати участь у виконанні деяких технологічних операцій або в управлінні комплексом.

Залежно від виду роботизованого виробничого процесу РТК можуть бути призначені для одержання заготовок, обробки деталей, виконання процесів складання або для реалізації контрольно-сортувальних і транспортно-перевантажувальних завдань, у тому числі для внутрішньоцехового транспортування і складських операцій.

При проектуванні різноманітних видів РТК, як правило, виділяють два етапи.

На першому етапі розглядають проблеми виробництва, вибирають об’єкти роботизації, склад основного технологічного устаткування, вид руху деталей, систему раціонального автоматизованого управління технологічним процесом і функціональними завданнями.

На другому етапі здійснюють безпосереднє проектування РТК, формують структуру, визначають кількість і характеристики промислових роботів і технологічного устаткування, розробляють раціональні планування устаткування РТК у виробничому приміщенні, вибирають компоновочні схеми РТК, складають і відпрацьовують алгоритми і програми системи управління РТК, що необхідні в період функціонування.

Компоновочні схеми РТК залежать від розв’язуваних технологічних завдань, рівня автоматизації, кількості і типу промислових роботів, їх технічних і функціональних можливостей. Розрізняють індивідуальне і групове обслуговування технологічного устаткування промислових роботів (ПР).

В умовах індивідуального обслуговування устаткування: ПР умонтований в одиницю технологічного устаткування; ПР розміщений поруч з одиницею технологічного устаткування; кілька ПР обслуговують одиницю технологічного устаткування.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи дистанційного голосового керування робототехнічним комплексом, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Dragon Naturally Speaking Preferred

Однозначно кращий з існуючих модулів розпізнавання мови. Весь алгоритм роботи із програмою гранично простий – підключаємо навушники й мікрофон до відповідних виходів з аудіоплати й запускаємо саму утиліту.

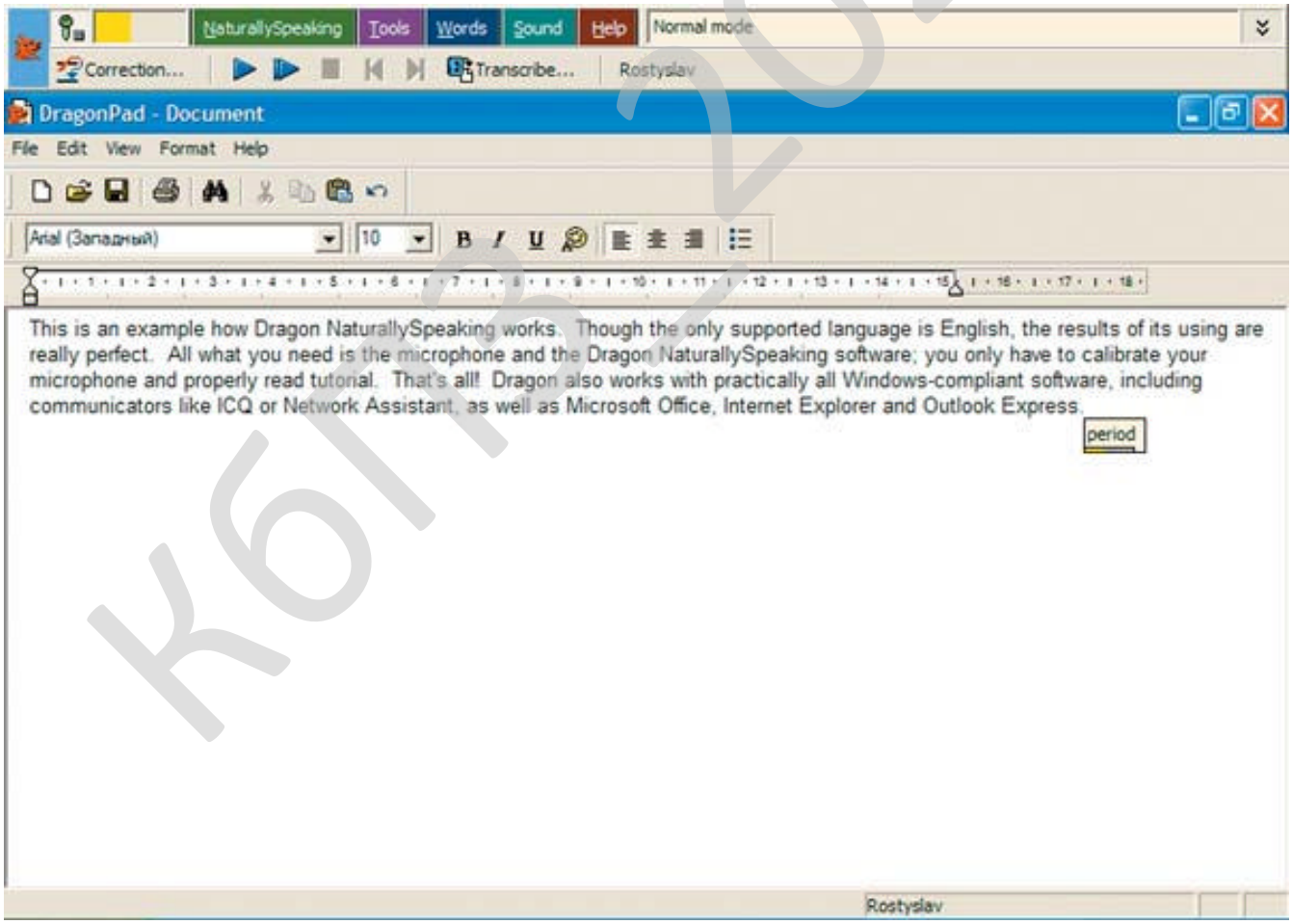

Рисунок 2.1 – Інтерфейс користувача Dragon Naturally Speaking Preferred

Спочатку користувачеві буде запропонований відкалібрувати рівень звуку з мікрофона й надиктувати комп'ютеру ряд уже готових текстів для більше тонкого підналаштування Dragon Naturally Speaking під ваші тембр, інтонацію й вимову. І нарешті, інтерактивний tutorial, де користувача навчають базовим голосовим командам.

Варто помітити, що ПК – це не живий співрозмовник і він не може додумати "проковтнуті" склади або зрозуміти нерозбірливо сказану пропозицію. Не менше значення має й власний акцент диктора – такий рівень англійського, котрий, наприклад, звучить на різноманітних міжнародних наукових конференціях, у принципі, непридатний для роботи. З іншого боку, завжди є можливість самонавчання: якщо Dragon ніяк не хоче розпізнавати якесь слово, не полінуєтеся заглянути в Lingvo і вимовити його з урахуванням правильної транскрипції. Запевняю вас, максимум за тиждень – два ви будете не тільки з легкістю надиктовувати кілобайти текстів, але й хизуватися серед знайомих істинно англійською вимовою.

Усе ще не задовольняє якість розпізнавання? Зверніться до Accuracy Center, що дозволить оптимізувати ваш користувальницький профіль і навчить, як поповнювати словник популярними неологізмами. Можливі й більше екзотичні дії начебто розпізнавання текстового вмісту wav-файлу (у тому числі й з Pocket PC або прямо з лінійного виходу аудіоплати). Крім того, Dragon Naturally Speaking уміє запускати різні програми, перемикатися між ними й навіть управляти поруч їхніх функцій (наприклад, починати/припиняти відтворення музики в медіапрогравачі або прямо працювати з меню). Ну а до складу версій Preferred і Professional додатково входить власний мовний движок Real-Speech 2, один з найбільш досконалих на сьогодні.

Але повернемося до запису мови. Особливо радує те, що надиктовувати текст можна не тільки в рідному текст-процесорі DragonPad, але й у будь-якому іншому аналогічному додатку – MS Word, Outlook Express, Internet Explorer і Corel WordPerfect. З таким же успіхом програма працює й з ICQ, мережним чатом

(Network Assistant) і іншими instant messengers; правда, тоді деякі команди становляться недоступні, зате для відправлення повідомлення навіть Enter натискати не треба, досить сказати: "New paragraph" – і ICQ автоматично зробить це. У більш спеціалізованих додатках, зокрема в тому же Word, застосовуються додаткові команди: форматування тексту, правопис, редагування – і все виключно за рахунок усного мовлення. Якщо ж стандартного набору наказів виявилося мало, завжди можна створити власні, тим самим ще більше розширивши функціональність Dragon. Варто небагато постаратися, і цілком реально набрати сторінку тексту без яких-небудь виправлень. Головне – вірне сполучення інтонації й, саме собою, вимови. Не розтягуйте фрази, але й не строчіть як з кулемета, інакше відсоток правильно зрозумілого матеріалу буде впевнено прагнути до нуля. Причому зовсім необов'язково постійно дивитися в словник – навіть якщо ви не зовсім вірно виговорили якесь словосполучення (наприклад, I'm very happy), відоме програмі, вона "догадається" автоматично виправити текст. Вражає? Вся справа у величезному словниковому запасі, що поряд із просунутою технологією розпізнавання мови не залишає ніяких шансів конкурентам.

Intelligent Voice Recognition System (IVOS)

Сама скромна (по розмірах дистрибутив1. програма в огляді виявила себе напрочуд гідно й значною мірою виправдала свою голосну назву. Причина тому – її універсальність, покликана повністю викорінити засобу "ручного" уведення інформації. Отже, IVOS дозволяє:

– розпізнавати мову й перетворювати її в текст у будь-якому Windows-сумісному тексті-процесорі;

– управляти своїм ПК за допомогою різноманітних голосових команд, а також створювати свої власні;

– озвучувати електронні книги за допомогою зовнішніх голосових движків.

Плюс, зрозуміло, такі дріб'язки, як витяг тексту з Wav-Файлів, зручна, не обтяжений екран панель керування програмою й демократична (у порівнянні з тим же Dragon) ціна. Після реєстрації користувачеві стає доступна технологія VoiceTouch, що дозволяє навчати ПК вашим власним усним наказам.

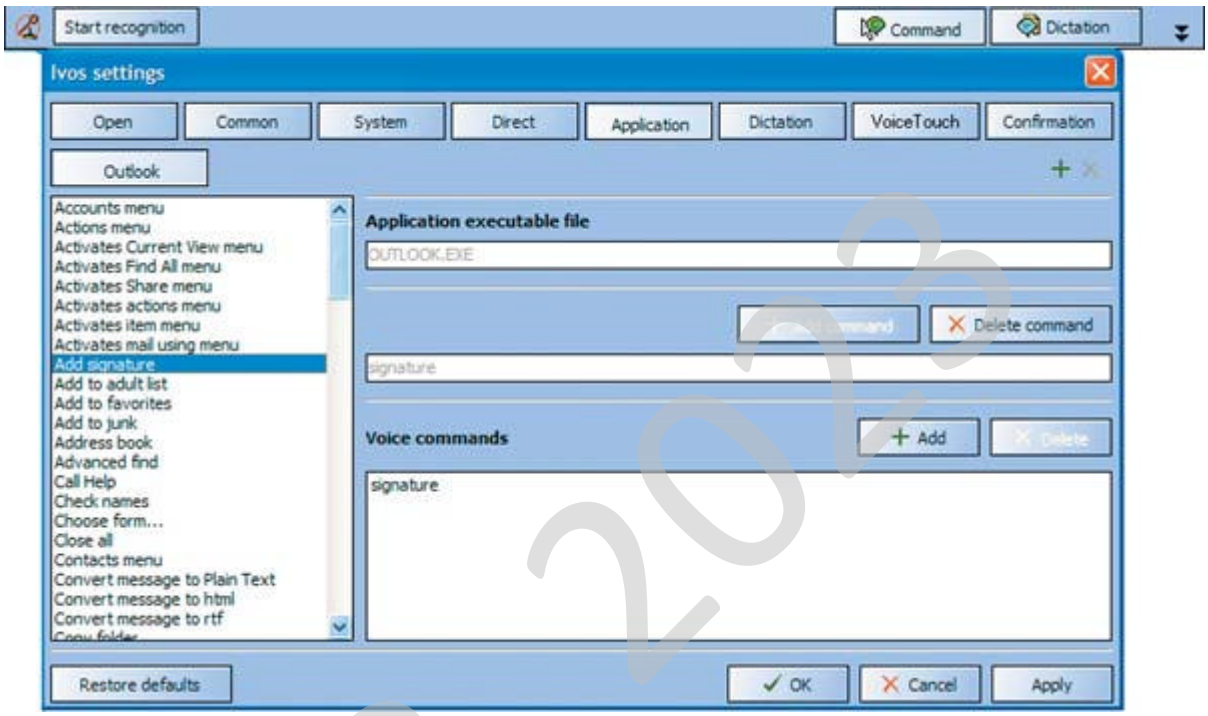

Рисунок 2.2 – Інтерфейс користувача Intelligent Voice Recognition System

Ефективність виконання команд напрочуд висока – мабуть, ще краще, ніж в Realize Voice. А от рівень розпізнавання "лекцій" буде нижче, що й не дивно: одна справа – зрозуміти пару слів, і зовсім інше – цільна пропозиція. Треба відзначити, що IVOS, як і багато інших програм розпізнавання мови, крім Dragon, використовує для таких цілей модуль Speech API від Microsoft, і її результативність у даній області прямо залежить від творчих успіхів цієї корпорації. Проте домогтися якісної роботи від IVOS можна вже зараз, начитавши програмі всі навчальні тексти, що є наявними у її запасі. Звичайно, до рівня Dragon Naturally Speaking у підсумку вона не дотягне, але набирати не занадто складні документи їй цілком під силу. А якщо ще регулярно поповнювати користувальницький словник, то й з науковими термінами не буде особливих

проблем. Правда, тут виникає дилема – за той тиждень, що прийдеться витратити на навчання утиліти всім тонкощам роботи з мовою, цілком можна ударними темпами опанувати методом сліпого десятипальцевого набору на клавіатурі... З іншого боку, кваліфікація користувача ПК лише підвищиться, якщо він буде володіти відразу декількома методами уведення інформації в комп'ютер.

Realize Voice

Realize Voice, на відміну від раніше розглянутого Dragon Naturally Speaking, не дуже-те здатна до стенографування (хоча така функція в її арсеналі і є), зате блискуче справляється з голосовими командами. Що примітно, винятково глибоких знань в області англійського не потрібно – завдяки розумному модулю евристичного аналізатора програма без особливих проблем знайде загальну мову практично з будь-яким диктором. Спектр функцій Realize Voice досить широкий: від запуску файлів, що виконуються, і ярликів програм до роботи з кореспонденцією й складними макросами. Як і в інших подібних програмах, від користувача потрібно лише підключений мікрофон і пара хвилин для того, щоб вникнути в курс справи. А перед тим, як приступитися до властиво спілкування з утилітою, варто позначити їй фронт робіт. За замовчуванням у цю категорію попадають ярлики системного меню, Робочого стола, уміст папки Вибране й панелі швидкого запуску, а також недавно відкриті документи й програми. Весь процес повністю автоматизований і виконується буквально миттєво. Правда, деякі незручності викликає неможливість використання в назві команд цифр – приміром, запустити DOOM 3 за допомогою голосового наказу вдасться, лише перейменувавши його ярлик в "DOOM Three". Те ж, до речі, стосується й кирилиці. Втім, у такому разі завжди можна вдатися до ручного наналаштування програми, прямо вказавши шлях до файлу, що цікавить вас / документу / графічному зображенню й т.д. Тут уже назва файлу і його координати ніякого значення не мають – будь він хоч абвгд.exe, та й Робочий стіл спотворювати не прийдеться.

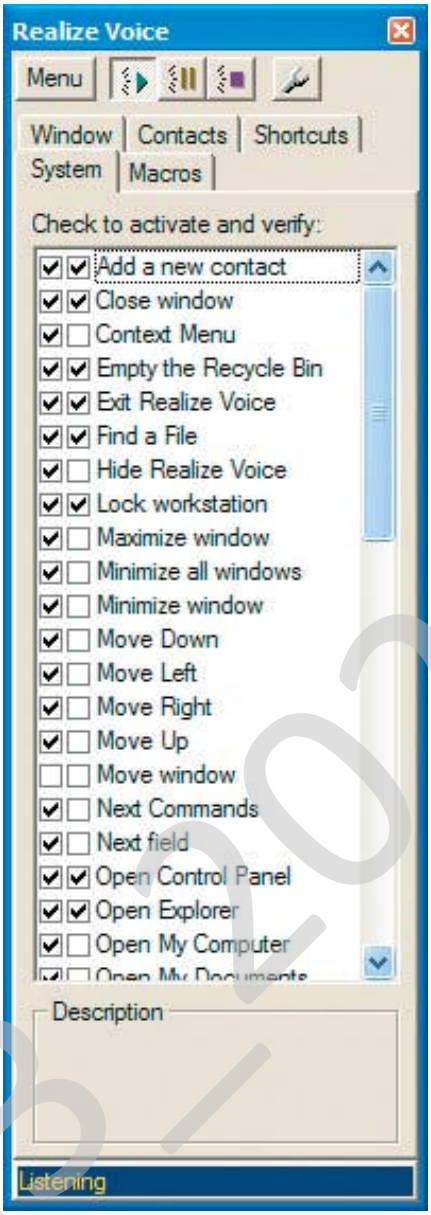

Рисунок 2.3 – Інтерфейс користувача Realize Voice

Досить порадував і набір убудованих системних команд для роботи з Windows – хоч він і не занадто великий, але переміщатися між відкритими вікнами, емулювати дію найпоширеніших клавіш (Spacebar, Insert, Home і т.д.), виключати й блокувати систему з його допомогою цілком реально.

Небагато про макроси. Утиліта дозволяє поєднувати під однією командою цілу серію операцій – починаючи від уведення символів із клавіатури й системних команд до синтезу мови за допомогою убудованого голосового движка. Правда, до такої ідилії, як запис CD за допомогою одного-єдиного словосполучення, поки

далеко, але час покаже... Головне, що вже зараз можна (і небезуспішно!) "покерувати" своїм домашнім вихованцем без усяких анахронізмів начебто миші й клавіатури.

Voice Studio

Мабуть, одна з деяких, якщо взагалі не єдина така програма, де наш віртуальний співрозмовник по ту сторону монітора нарешті-те знайшов матеріальну форму. І хоча технологію MS Agent, що використовується для даних цілей, поки важко назвати прообразом штучного інтелекту, всі передумови для цього в неї є. Анімований помічник не тільки наділений деякою часткою самостійності, але й уміє відповідати на ряд стандартних фраз (начебто "Hello!", "How do you feel", "Bad computer" і т.д.). При бажанні його словниковий і фразеологічний запас легко поповнити, а крім того, задати його дії залежно від "настрою". Хоча подібна балаканина із ПК і буде обмежена рамками знань програми, ніхто не заважає розширити їх практично нескінченно.

Властиво з функціональністю Voice Studio усе в повному порядку – стенографування (правда, Dragon значно краще), різноманітні голосові команди (для більшої зручності й найшвидшого запам'ятовування їх можна роздрукувати), а також прийнятний машинний синтез мови. З більше серйозних речей – створення макросів для запуску відразу серії операцій за допомогою одного ключового слова, навіть запис і відтворення рухів миші! Нагадаю, що остання "фіча" широко використовується в багатьох альтернативних браузерах начебто GreenBrowser або MyIE2 для виконання ряду дій (перехід на іншу сторінку, відкриття нового вікна й т.д.). Тепер не треба ніяких непотрібних дій – досить вимовити відповідну команду, і комп'ютер автоматично відновить записаний раніше скрипт.

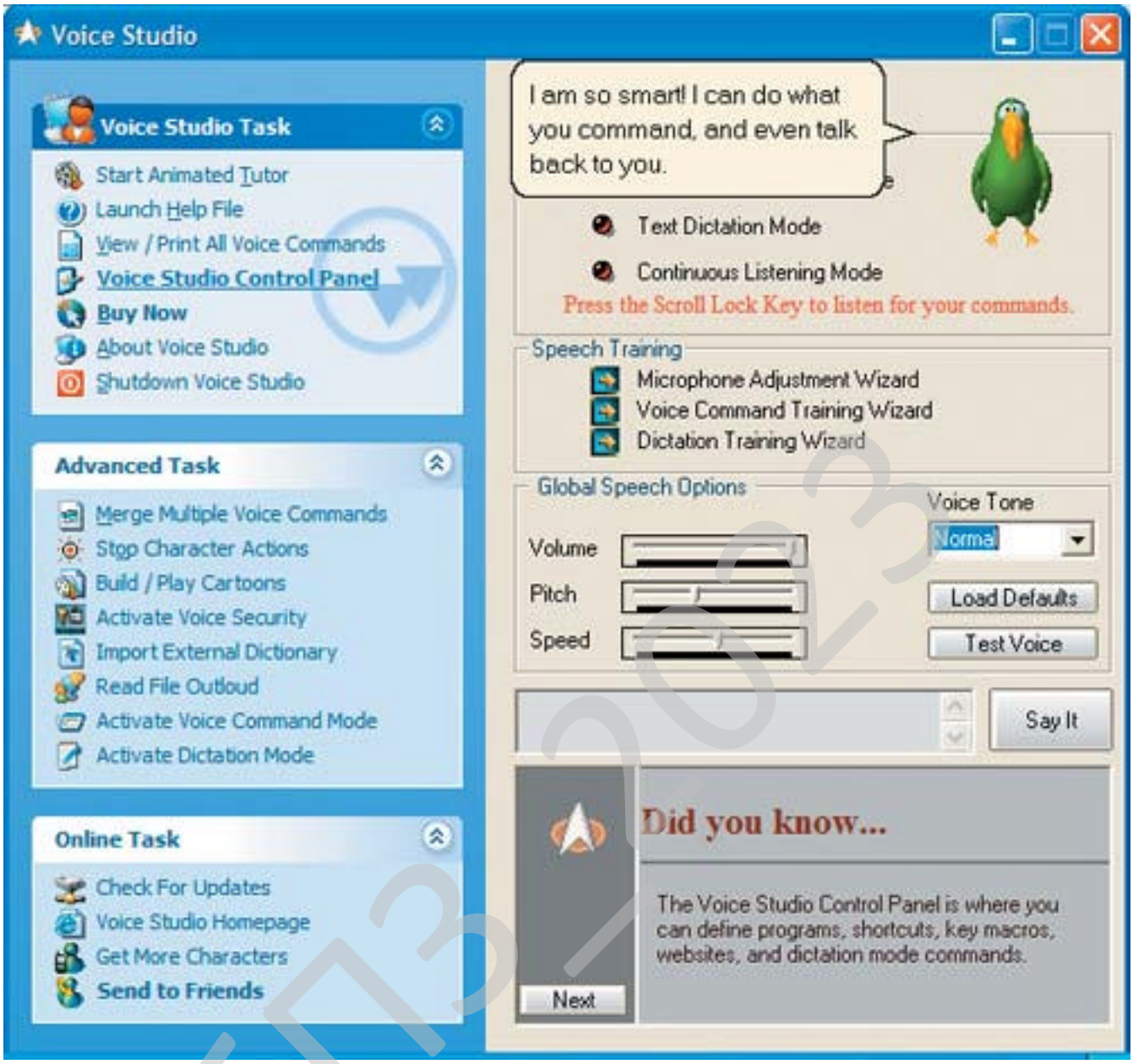

Рисунок 2.4 – Інтерфейс користувача Voice Studio

А поки Voice Studio за приголомшливу дружність і легкість у роботі безсумнівно заслуговує вищої оцінки. Нехай коректний запис мови їй поки не під силу, але керування ПК голосом тут просто незрівнянно. Краща з таких утиліт і гідне доповнення до Dragon.

Dictation

Незважаючи на, здавалися б, зовсім стандартні базові вміння, дечим Dictation все-таки похвастатися може. У першу чергу, це технологія Point-and-

Speak, що дозволяє з легкістю створювати команди для уведення паролів, запуску ПЗ й диктувати практично у всіх Windows-додатках. Заявлена інтеграція з MS Word, а також інтелектуальна технологія правильного визначення фраз. Правда, реалізована вона надзвичайно незручно – у вигляді спливаючого вікна, що з'являється при кожному сказаному слові й лише відбиває всяке бажання працювати.

Рисунок 2.5 – Інтерфейс користувача Dictation

Добре хоч, що її можна відключити. Dictation використовує все той же SAPI, так що якість її принципово не відрізняється від іншого ПЗ, заснованого на тій же технології (Voxx, IVOS, Realize Voice та ін.). З додаткових функцій варто відзначити WAV Recorder для захоплення інформації з аудіокасет, мобільних пристроїв, мікрофонів і наступного запису її в wav-файли; потім текст із них

витягає за допомогою окремого апплету Dictation – Wave-to-Text. Поки що він, звичайно, усе ще далекий від ідеалу, але якщо в диктора чітка мова й непогана вимова, то проблем не виникне.

Voxx

Ще один "майстер на всі руки", що дозволяє досхочу побалакати із ПК. Список можливостей програми досить нагадує такий в IVOS (стенографування/голосові команди/читання тексту), за винятком того, що тут є корисний бонус – скрупульозне озвучування кожної вашої дії, будь те набір тексту або відкриття файлу.

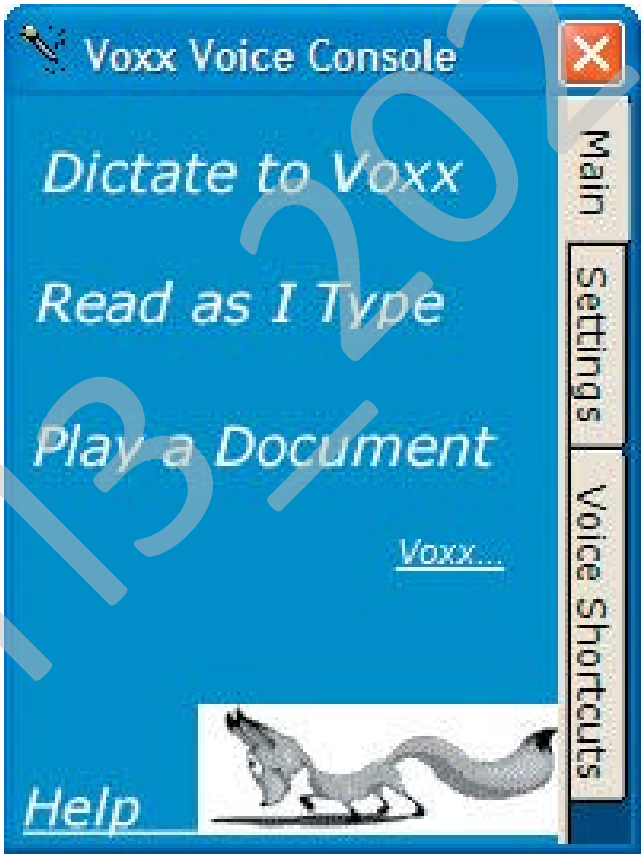

Рисунок 2.6 – Інтерфейс користувача Voxx

Програма використовує той же Microsoft Speech API, що й IVOS, тому і якість розпізнавання в неї аналогічне. Є в наявності непоганий набір голосових команд для навігації браузером, елементарних операцій у текстовому редакторі (cut/copy/paste і т.д.), а також роботи з вікнами, є ярлики виклику системних

аплетів, навіть відкриття/закриття лотка оптичного приводу – загалом, усе для комфортної роботи. Що ж стосується синтезу мови, то він прямо залежить від відповідних модулів, установлених у системі. Безкоштовні движки від Microsoft, що поставляються разом із програмою, далекі від ідеалу, але, у принципі, до них звикнути можна. Більше зручний варіант, на жаль, не безоплатний – спробувати сторонні розробки, зокрема Digit PC, до всього іншого який володіє досить непоганим російськомовним диктором. З огляду на всі плюси й мінуси, Voxx буде непоганим кандидатом на покупку. До речі, trial-версія обмежена лише кількістю фраз/команд на сеанс роботи; для початку нового сеансу досить запустити знову програму.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для реалізації програми мною була використана мова програмування Visual C++. У зв'язку з тим, що сьогодні рівень складності програмного забезпечення дуже високий, розробка застосунків Windows з використанням тільки якої-небудь мови програмування значно утрудняється. Програміст повинен затратити масу часу на рішення стандартних завдань по створенню багатовіконного інтерфейсу. Реалізація технології зв'язування й вбудовування об'єктів – OLE – зажадає від програміста ще більш складної роботи. Щоб полегшити роботу програміста практично всі сучасні компілятори з мови C++ містять спеціальні бібліотеки класів. Такі бібліотеки містять у собі практично весь програмний інтерфейс Windows і дозволяють користуватися при програмуванні засобами більш високого рівня, чим звичайні виклики функцій. За рахунок цього значно спрощується розробка застосунків, що мають складний інтерфейс користувача, полегшується підтримка технології OLE і взаємодія з базами даних. Сучасні інтегровані засоби розробки застосунків Windows дозволяють автоматизувати процес створення застосунка. Для цього використовуються генератори застосунків. Програміст відповідає на питання

генератора застосунків і визначає властивості застосунка – чи підтримує воно багатовіконний режим, технологію OLE, тривимірні органи керування, довідкову систему. Генератор застосунків, створить додаток, що відповідає вимогам, і надасть вихідні тексти. Користуючись їм як шаблоном, програміст зможе швидко розробляти свої застосунки. Подібні засоби автоматизованого створення застосунків включені в компілятор Microsoft Visual C++ і називаються MFC AppWizard. Заповнивши кілька діалогових панелей, можна вказати характеристики застосунка й одержати його тексти, постачені великими коментарями. MFC AppWizard дозволяє створювати одновіконні й багатовіконні застосунки, а також застосунки, що не мають головного вікна, – замість нього використовується діалогова панель. Можна також включити підтримку технології OLE, баз даних, довідкової системи. Звичайно, MFC AppWizard не всесильний. Прикладну частину застосунка програмістові прийдеться розробляти самостійно. Вихідний текст застосунка, створений MFC AppWizard, стане тільки основою, до якої потрібно підключити інше. Але працюючий шаблон застосунка – це вже половина всієї роботи. Вихідні тексти застосунків, автоматично отриманих від MFC AppWizard, можуть становити сотні рядків тексту. Набір його вручну був би дуже стомлюючий. Потрібно відзначити, що MFC AppWizard створює тексти застосунків тільки з використанням бібліотеки класів MFC (Microsoft Foundation Class library). Тому тільки вивчивши мову C++ і бібліотеку MFC, можна користуватися засобами автоматизованої розробки й створювати свої застосунки в найкоротший термін. Як уже згадувався, MFC – це базовий набір (бібліотека) класів, написаних мовою C++ і призначених для спрощення й прискорення процесу програмування для Windows. Бібліотека містить багаторівневу ієрархію класів, що нараховує близько 200 членів. Вони дають можливість створювати Windows-застосунки на базі об’єктно-орієнтованого підходу. З погляду програміста, MFC являє собою каркас, на основі якого можна писати програми для Windows. Бібліотека MFC розроблялася для спрощення завдань, що стоять перед програмістом. Як відомо, традиційний метод програмування під Windows вимагає написання досить довгих і складних програм, що мають ряд специфічних особливостей. Зокрема, для створення тільки каркаса програми таким методом

знадобиться близько 75 рядків коду. У міру ж збільшення складності програми її код може досягати воістину неймовірних розмірів. Однак та ж сама програма, написана з використанням MFC, буде приблизно в три рази менше, оскільки більшість приватних деталей приховано від програміста.

Одною з основних переваг роботи з MFC є можливість багаторазового використання того самого коду. В зв'язку з тим, що бібліотека містить багато елементів, загальних для всіх Windows-застосунків, немає необхідності щораз писати їх заново. Замість цього їх можна просто успадковувати (говорячи мовою об'єктно-орієнтованного програмування). Крім того, інтерфейс, забезпечуваний бібліотекою, практично незалежний від конкретних деталей, його що реалізують. Тому програми, написані на основі MFC, можуть бути легко адаптовані до нових версій Windows (на відміну від більшості програм, написаних звичайними методами). Ще однією істотною перевагою MFC є спрощення взаємодії із прикладним програмним інтерфейсом (API) Windows. Будь-який додаток взаємодіє з Windows через API, що містить кілька сотень функцій. Значний розмір API утрудняє спроби зрозуміти й вивчити його цілком. Найчастіше навіть складно простежити, як окремі частини API зв'язані один з одним! Але оскільки бібліотека MFC поєднує (шляхом інкапсуляції) функції API у логічно організовану безліч класів, інтерфейсом стає значно легше управляти.

Оскільки MFC являє собою набір класів, написаних мовою C++, тому програми, написані з використанням MFC, повинна бути в той же час програмами на C++. Для цього необхідно володіти відповідними знаннями. Для початку необхідно вміти створювати власні класи, розуміти принципи спадкування й уміти перевизначати віртуальні функції. Хоча програми, що використовують бібліотеку MFC, звичайно не містять занадто специфічних елементів з арсеналу C++, для їхнього написання проте потрібні солідні знання в даній області.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи дистанційного голосового керування робототехнічним комплексом.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Метод розрахунку ознак мовного сигналу – ЛСК

Мовний сигнал описується в термінах лінійних дискретних систем зі змінними параметрами й передатною функцією в частотній області виду**:**

$$H(z) = \frac{S(z)}{U(z)} = G \cdot \frac{1 + \sum_{l=1}^{q} b_l \cdot z^{-l}}{1 + \sum_{k=1}^{p} a_k \cdot z^{-k}}. \quad (3.1)$$

Найбільше широко для опису мовного сигнал (МС) застосовується полюсна модель лінійного пророкування, що представляється у вигляді:

$$H(z) = \frac{1}{A(z)} = G \frac{1}{1 + \sum_{i=1}^{N} a_i z^{-i}}, \quad (3.2)$$

де N – порядок моделі.

Параметрами такої моделі є коефіцієнти лінійного пророкування $\{a\}$, що обчислюються на кожному кадрі мовного сигналу, або еквівалентні їм параметри – ЛСК, запропоновані Ітакурой.

Корінь у загальному випадку можуть бути отримані в результаті рішення двох рівнянь:

$$\mathrm{Re}\{z^R A_N(z)\}_{z=e^{j\hat{\omega}}} = 0,\ \mathrm{Im}\{z^R A_N(z)\}_{z=e^{j\hat{\omega}}} = 0 \text{ при } R \geq (N/2), \quad (3.3)$$

де $A_N(z) = 1 + \sum_{i=1}^{N} a_i z^{-i}$.

При цьому на підставі нової теорії ЛСК, запропонованої А.А. Ланне, коріння можуть розраховуватися по-різному залежно від параметра R. У рамках цієї теорії виділено кілька приватних випадків розрахунку ЛСК.

Модель розрахунку ЛСК для R = N

У даній роботі розглядається випадок, коли $R = N = 10$. Досить вирішити тільки одне рівняння порядку *N*, щоб по його корінням знайти всі коефіцієнти вихідного багаточлена.

Задається порядок моделі (ступінь апроксимуючого полінома) *ORD*. На вхід надходить відрізок сигналу (кадр) тривалості *FRM*:

$$SG = \{sg_0, sg_1, ..., sg_{FRM}\}. \quad (3.4)$$

Для усунення граничних ефектів виробляється згладжування ваговою функцією Хеммінга:

$$sh_i = sg_i \cdot (0{,}54 + 0{,}46 \cdot \cos(\frac{2\pi i}{FRM - i})), \quad (3.5)$$

де $i = 0...FRM$.

Виконується розрахунок коефіцієнтів передатної функції за допомогою методу найменших квадратів і алгоритму Левинсона-Дарбіна.

Первинна ініціалізація:

$$E_0 = \sum_{i=1}^{FRM} sg_i{}^2. \quad (3.6)$$

У циклі від 1 до *ORD* виробляються наступні обчислення:

- Обчислення коефіцієнта автокореляції:

$$R_i = \sum_{l=1}^{FRM} sg_l sg_{l-i}. \quad (3.7)$$

- Обчислення коефіцієнта відбиття:

$$r_i = \frac{R_i - \sum_{k=1}^{i} a_k{}^{(i-1)} R_{i-k}}{E_{i-1}}. \quad (3.8)$$

- Завдання первісного наближення:

$$a_i^{(i)} = r_i . \quad (3.9)$$

– Уточнення значень коефіцієнтів:

$$a_k^{(i)} = a_k^{(i-1)} - r_i a_{i-k}^{(i-1)}, \quad (3.10)$$

де $1 \le k \le i-1$.

– Обчислення поточної помилки пророкування:

$$E_i = (1 - r_i^2) E_{i-1} . \quad (3.11)$$

– На останньому кроці циклу виходить остаточне рішення:

$$a_k = a_k^{(k)}, \text{ де } 1 \le k \le i-1. \quad (3.12)$$

Далі розрахунок коефіцієнтів відбиття по формулах кратних дуг:

$$G = \{g_0, g_1, ..., g_{ORD}\}. \quad (3.13)$$

Пошук корінь полінома методом Ньютона:

$$G(x) = \sum_{i=0}^{ORD} g_i x^i . \quad (3.14)$$

Розрахунок набору ЛСК:

$$w_i = \arccos(x_i), \quad (3.15)$$

де $i = 0...ORD-1$.

Використання ЛСК як інформативні ознаки МС

При розрахунку ЛСК на тривалому МС (рис. 3.1), виробляється його розбивка на кадри з перекриттям. У результаті розрахунків виходить набір значень ЛСК (рис. 4).

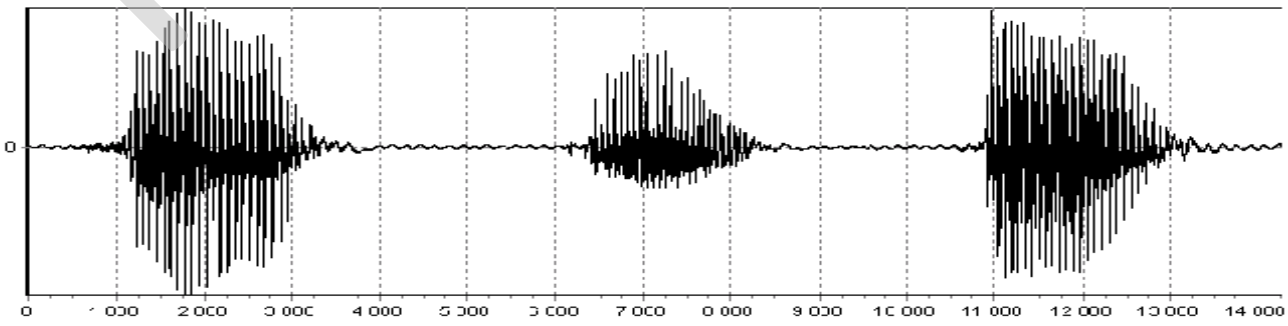

Рисунок 3.1 – Часова діаграма голосних фонем «а», «і», «о»

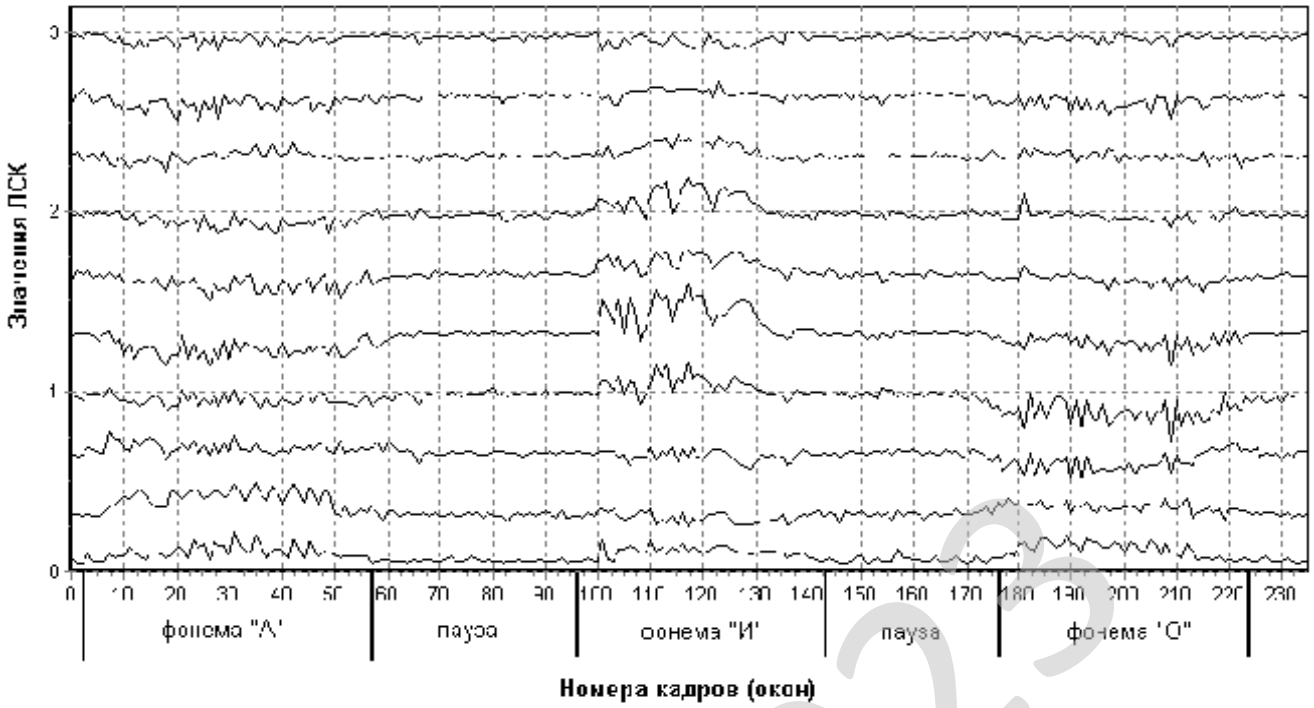

Рисунок 3.2 – Набір ЛСК для трьох фонем (порядок моделі – 10 корінь)

На рис 3.2 спостерігається порушення певних корінь при проголошенні фонем. Це обумовлено тим, що ЛСК несуть у собі спектральну інформацію про МС. Порушення корінь відбувається в області формантних частот голосних звуків.

Значення кожного ЛСК використовується як координата в N-мірному просторі ознак. На рис 3.3 і 3.4 показані образи трьох фонем у двомірному й тривимірному підпросторах. Сполучні лінії між точками відображають послідовність кадрів МС. Для деяких комбінацій ЛСК спостерігається впевнений поділ фонем – точки групуються в межах однієї області. Ця властивість дозволяє використовувати ЛСК як інформативні ознаки в СРМ.

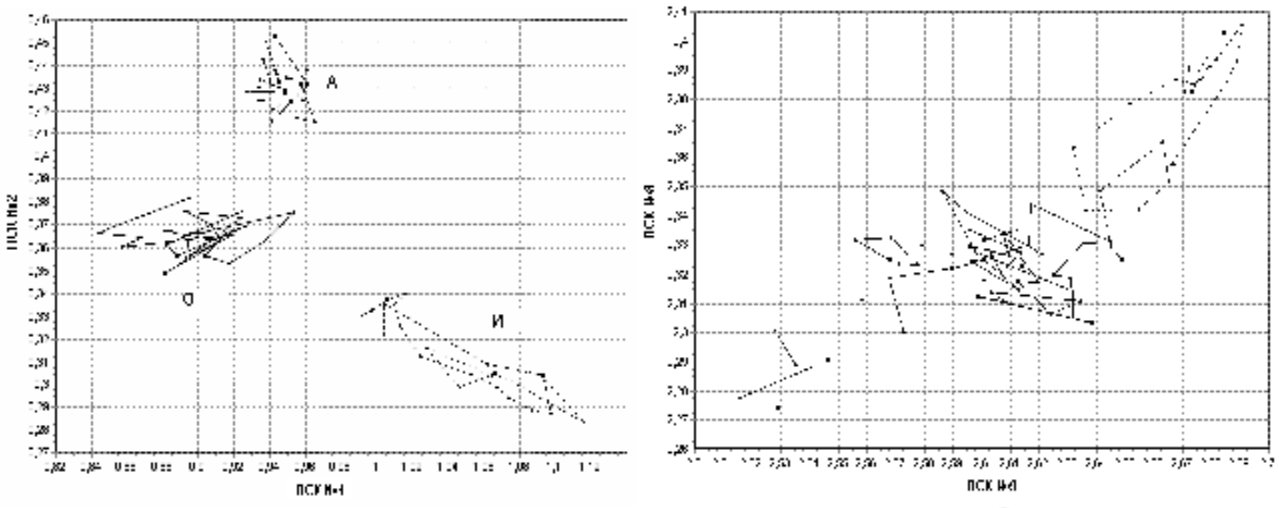

Рисунок 3.3 – Образи фонем у двомірному підпросторі ознак ЛСК

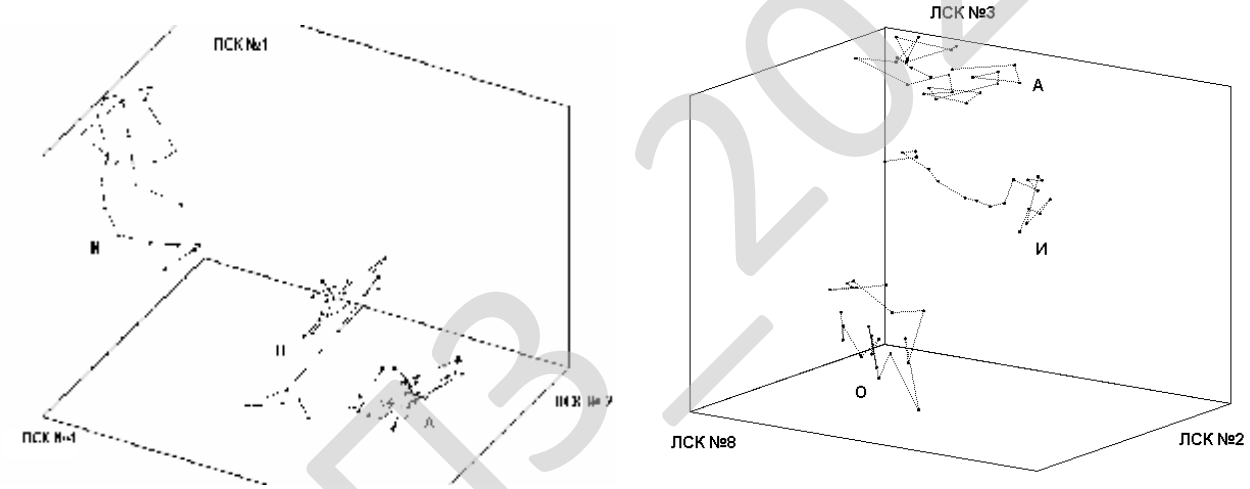

Рисунок 3.4 – Образи фонем у тривимірному підпросторі ознак ЛСК

Далі розглянемо моделі побудови словників еталонів, методики пошуку по них, проводиться критерій для оцінки вірогідності розпізнавання мовної команди.

Вибір методики формування словника еталонів

Розпізнавання мови шляхом виділення окремих фонем на практиці не принесло істотних результатів. Якщо повернутися до проблеми сприйняття мови людиною, то виявляється, що навіть досвідчені фонетисти із працею справляються із завданням розчленовування злитої мови на короткі сегменти.

Найчастіше щоб розпізнати окрему фонему, слухачеві необхідно почути слово цілком або навіть трохи рядом вартих слів.

Відомо, що чим триваліше мовна одиниця, тим краще вона сприймається на слух. Виходячи із цього, для системи розпізнавання мовних команд як еталони найбільше доцільно використовувати цілі слова.

На рис 3.5 і 3.6 показані два слова, записані від різних дикторів. Слова представлені у вигляді точок у підпросторі двох ЛСК. Очевидно, що окремі фонеми досить важко виділити із цілого слова. Сполучні лінії (траєкторії точок ЛСК) відображають перебудування голосового тракту людини в процесі проголошення звуків. Для тих самих слів траєкторії візуально схожі. Ця властивість дозволяє використовувати набори векторів ЛСК, з обліком їхньої часової послідовності, як елементи навчальних словників.

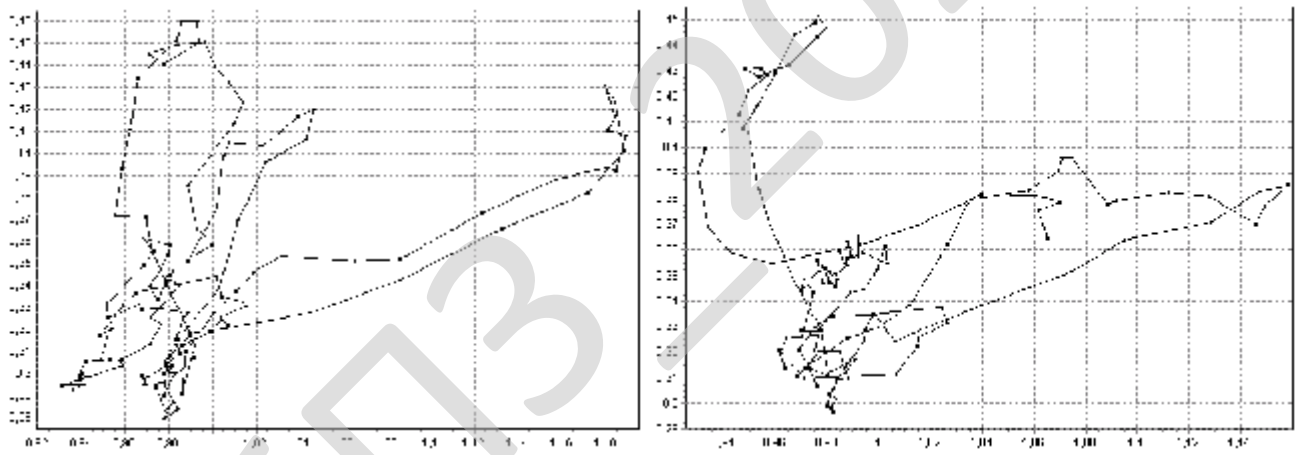

Рисунок 3.5 – Образи слова «повідомлення» для двох різних дикторів

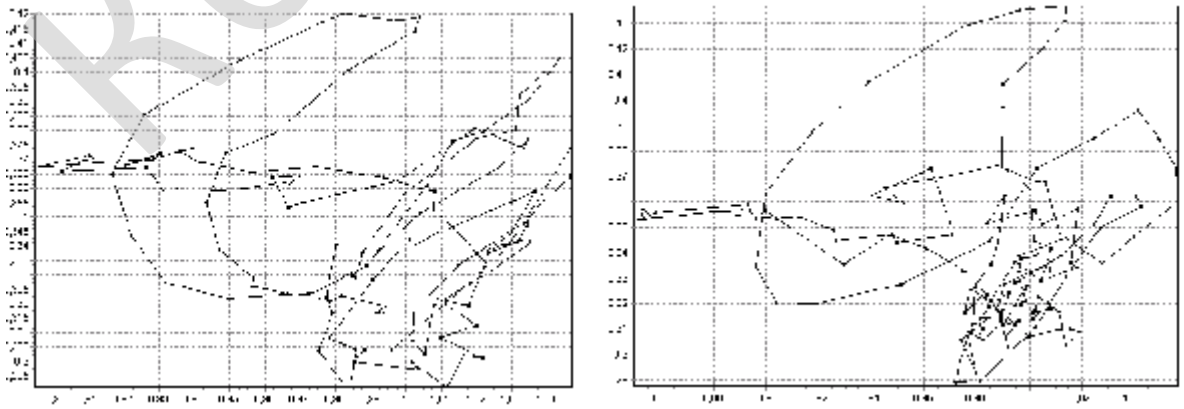

Рисунок 3.6 – Образи слова «наналаштування» для двох різних дикторів

Оцінка міри близькості між вхідним МС і еталоном виробляється за допомогою методу нелінійного часового вирівнювання (динамічного програмування). Це один з найбільш потужних і широко відомих математичних методів сучасної теорії керування, був запропонований наприкінці 50-х років американським математиком Р. Беллманом для рішення оптимізаційних завдань. Метод дозволяє порівнювати різні по тривалості зразки. Застосовно до мовних сигналів це означає, що порівняння з еталонами можливо практично незалежно від темпу мови.

Нехай рівняється два зразки сигналів, представлених у вигляді масиву векторів (для МС це набори ЛСК):

$$X = \{\bar{x}_0, \bar{x}_1, ..., \bar{x}_i, ..., \bar{x}_N\} \text{ та } Y = \{\bar{y}_0, \bar{y}_1, ..., \bar{y}_i, ..., \bar{y}_M\} \quad (3.16)$$

Розходження між векторами двох образів визначається послідовністю станів C_K і позначається:

$$F() = C_0, C_1, ..., C_k, ..., C_K, \quad (3.17)$$

де C_0 й C_K – початковий і кінцевий стани, $F()$ – функція часового вирівнювання, що проектує часову область одного образа на часову область іншого образа.

Метод ДП полягає в тім, що шукається така функція $F()$, при якій шлях зі стану C_0 в стан C_K, є оптимальним, тобто буде отримана мінімальна накопичена відстань між двома образами.

При побудові оптимального шляху, на кожному кроці алгоритму використовується основна формула ДП:

$$d_{i,j} = \min\begin{Bmatrix} d_{i,j-1} + r(\bar{x}_i, \bar{y}_j) \\ d_{i-1,j-1} + r(\bar{x}_i, \bar{y}_j) \\ d_{i-1,j} + r(\bar{x}_i, \bar{y}_j) \end{Bmatrix}, \text{ де } i = 0...N, j = 0...M. \quad (3.18)$$

Як відстань між векторами використовується зважена евклідова метрика:

$$r(\bar{x}, \bar{y}) = \sum_{k=0}^{N\_SEC-1} (x_k - y_k)^2, \quad (3.19)$$

де N_SEC – розмірність векторів ознак.

На виході процедури порівняння виходить деяке число (міра близькості), що представляє собою величину, зворотну ступені близькості між сигналами.

Процедура пошуку по словнику полягає в послідовному порівнянні вхідного сигналу з кожним з еталонів мовних команд. У табл. 3.1 показаний результат пошуку команди «повідомлення» у словнику із чотирьох командних слів. У результаті вхідний сигнал правильно розпізнаний системою. На рис 3.9 відображаються траєкторії найкоротших переходів по кадрам від еталонних сигналів до розпізнаваного. Дані по осі ординат нормовані по тривалості еталонних сигналів. По осі абсцис ідуть номера кадрів вхідного сигналу. Ділянки із крутими переходами між точками відображають автоматичне часове масштабування сигналів. Це відбувається, наприклад, якщо при проголошенні диктором розтягується голосний звук.

Таблиця 3.1 – Результат пошуку команди «повідомлення» у словнику із чотирьох командних слів

Еталон командного слова	Міра близькості
Повідомлення	1,85
Журнал	5,13
Диспетчер	6,82
Календар	4,33

Ідеальний випадок, коли розпізнаваний сигнал збігається з еталонним, являє собою діагональну східчасту траєкторію з лівого нижнього кута у верхній правий. На рис 3.9 для еталонів «журнал» і «календар» спостерігається істотне відхилення від діагоналі, що може бути додатковим критерієм для ухвалення рішення при розпізнаванні слів.

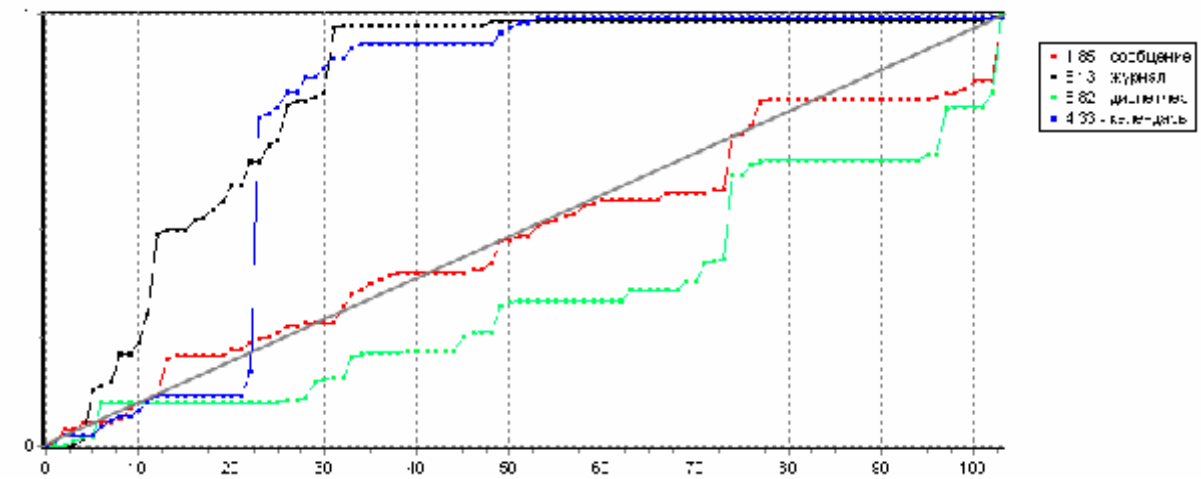

Рисунок 3.7 – Оптимальні траєкторії при порівнянні з еталонами

Перше ніж буде розпізнане ціле командне слово, на базі запропонованої моделі можливе **розпізнавання більше дрібних мовних одиниць**. Це дозволить скоротити область пошуку в словнику й підвищити точність алгоритму. На рис 3.11 представлений результат розпізнавання цілого слова «режими» на словнику, що складається з набору складів. У якості одного з елементів словника використовується «еталон тиші» (позначений як «_»), що дозволяє без застосування додаткових алгоритмів виділяти паузи в мовних сигналах.

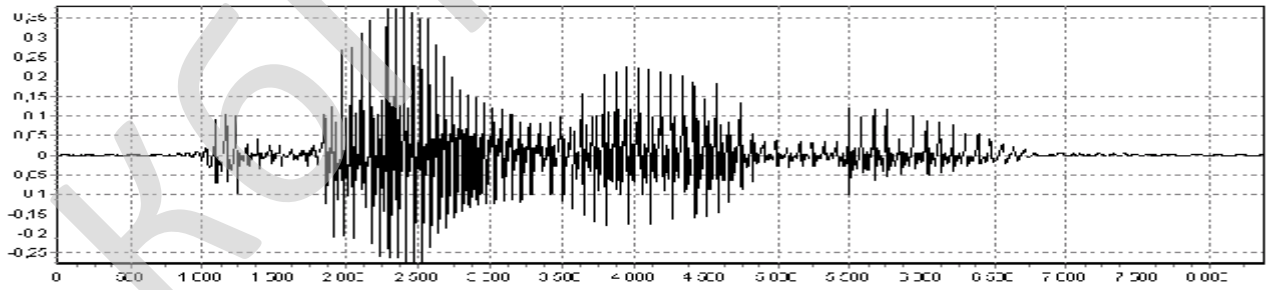

Рисунок 3.8 – Часова діаграма слова « ре-жи-ми»

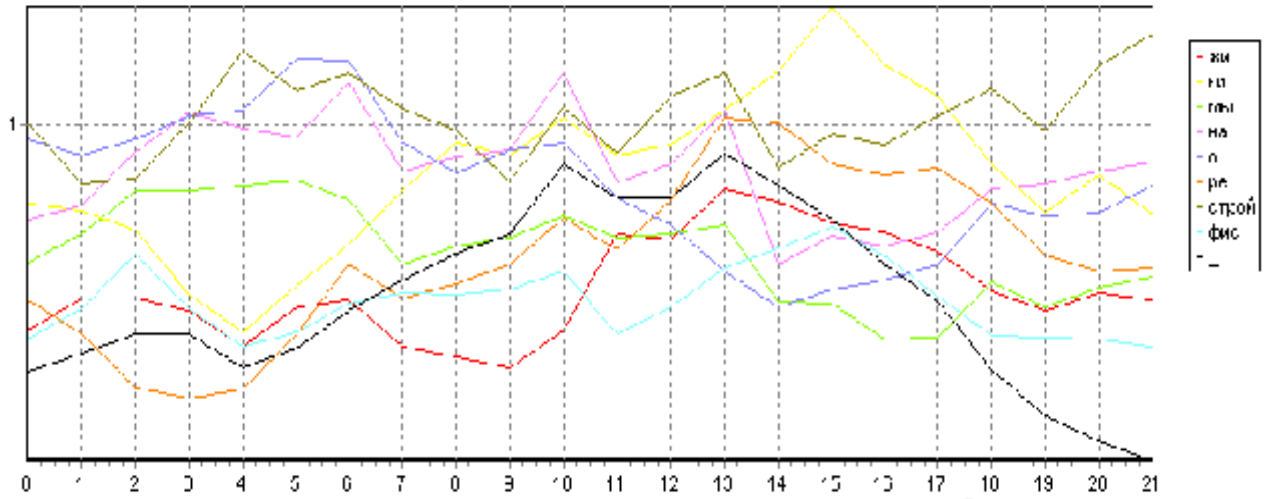

Рисунок 3.9 – Результат пошуку складів: «__ререре__жижижижи_мимимими____»

Вхідний сигнал розбивається на кадри по середній довжині еталонів. На графіку показані діаграми міри близькості до кожного з еталонів для всіх кадрів мовного сигналу. У результаті одержуємо послідовність розпізнаних складів. Шляхом згортки й подальша семантична обробка можлива одержання цілого слова. Дана методика може використовуватися для побудови СРМ на словниках більших обсягів.

Запропоновано рішення завдання пошуку слів у безперервному мовному потоці. Як елементи словника використовуються цілі слова. На вхід системи подається тривала ділянка мовного сигналу. У даному прикладі, фраза: «Чорна тойота номер три два один убік Києва» (рис 3.10).

Пошук іде без попередньої сегментації фрази на окремі слова. На рис 3.10 і 3.11 спостерігаються локальні мінімуми в області шуканих еталонних одиниць. На рис 3.12 яскраво вираженого мінімуму ні, тому що шукане слово («зелена») не було вимовлено в пропозиції. Співвідношення значення середньої міри близькості по всім кадрам МС і значення міри близькості на локальному мінімумі є критерієм, що дозволяє автоматично визначати, є присутнім чи взагалі шукане слово в аналізованій фразі.

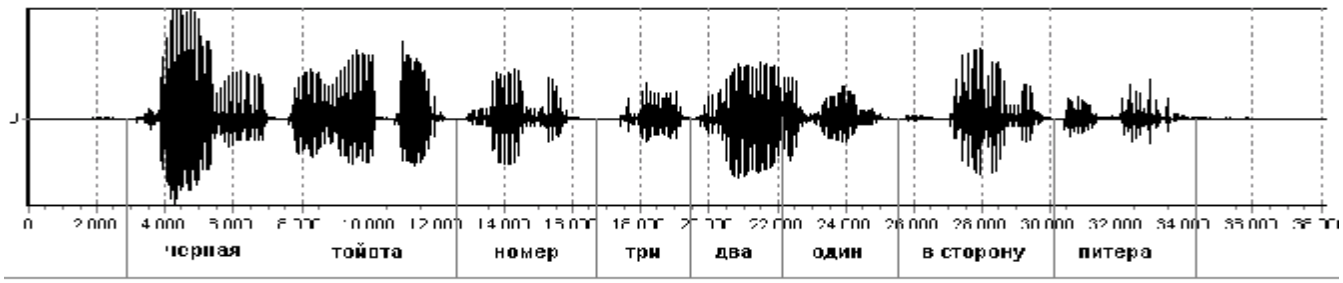

Рисунок 3.10 – Часова діаграма цілої фрази

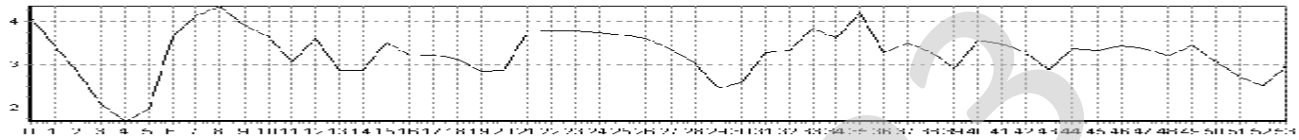

Рисунок 3.11 – Пошук слова «чорна» (співвідношення міри близькості = 0,5)

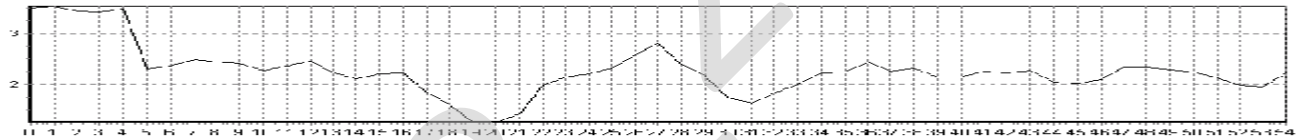

Рисунок 3.12 – Пошук слова «номер» (співвідношення міри близькості = 0,5)

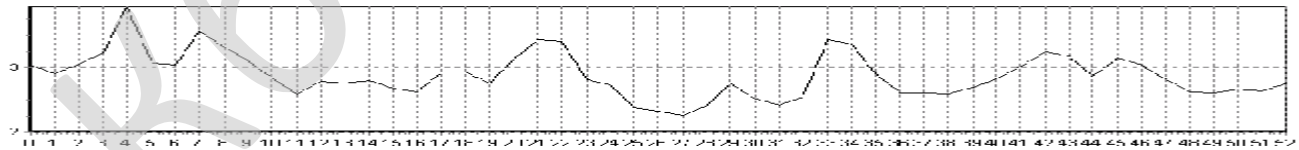

Рисунок 3.13 – Пошук слова «зелена» (співвідношення міри близькості = 0,8)

Критерій для оцінки вірогідності розпізнавання слів

При розпізнаванні мовних команд на базі словника з набору цілих слів, виходить таблиця зі значеннями міри близькості до елементів словника. Еталон з мінімальним значенням є шуканим – розпізнаним. Навіть якщо на вхід системи буде подане слово, що не входить у словник, у кожному разі буде отриманий результат – один з еталонів. Що приведе до помилки розпізнавання.

Запропоновано рішення завдання автоматичного відсівання помилкових спрацьовувань системи. Таблиця результатів розпізнавання нормується (табл. 3.2). Далі підраховується різниця в значенні міри близькості між першим і другим еталоном. У даному прикладі це 0,73. Якщо ця різниця не перевищує граничне значення 0,5, то слово буде вважатися нерозпізнаним і системою буде виданий запит на повторне уведення команди. Запропонований критерій дозволяє оцінювати вірогідність розпізнавання поточного слова.

Таблиця 3.2 – Таблиця результатів розпізнавання

Еталон	Міра близькості	Після нормування
Повідомлення	1,74	1,00
Пам'ять	3,01	1,73
Наналаштування	3,06	1,75
Годинники	3,45	1,98
Офіс	3,53	2,03
Режими	3,68	2,11
Засоби	4,06	2,33
Контакти	4,13	2,37
Теми	4,15	2,38
Журнал	4,25	2,44
Зв'язок	4,58	2,63
Календар	4,70	2,70

Оцінка впливу параметрів моделі ЛП на вірогідність розпізнавання

У ході досвідів, на словнику з 42 командних слів від 4 дикторів, варіювався розмір кадрів МС і ступінь апроксимуючого полінома (порядок моделі). На рис 3.14 і 3.15 наведені графіки відповідних залежностей. Найкраща вірогідність розпізнавання досягається, коли розмір вікна збігається з періодами основного тону МС. При зміні порядку моделі, максимум досягається на 10 коріннях, далі спостерігається пологий графік кривої.

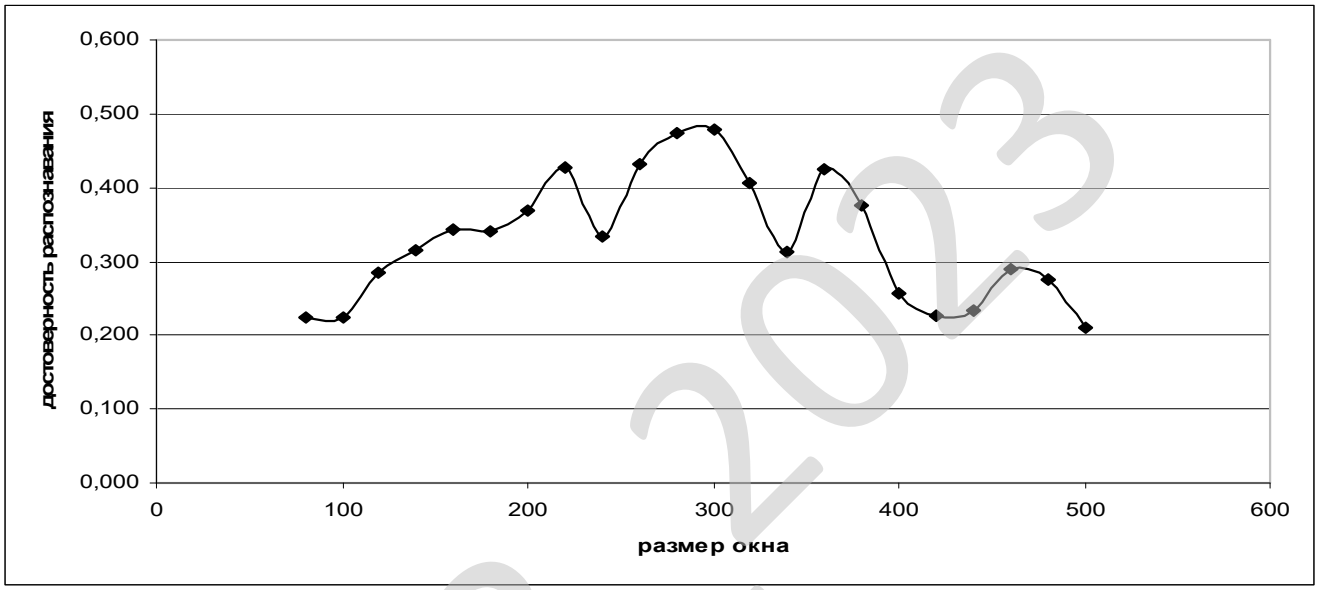

Рисунок 3.14 – Вплив розміру вікна на вірогідність розпізнавання

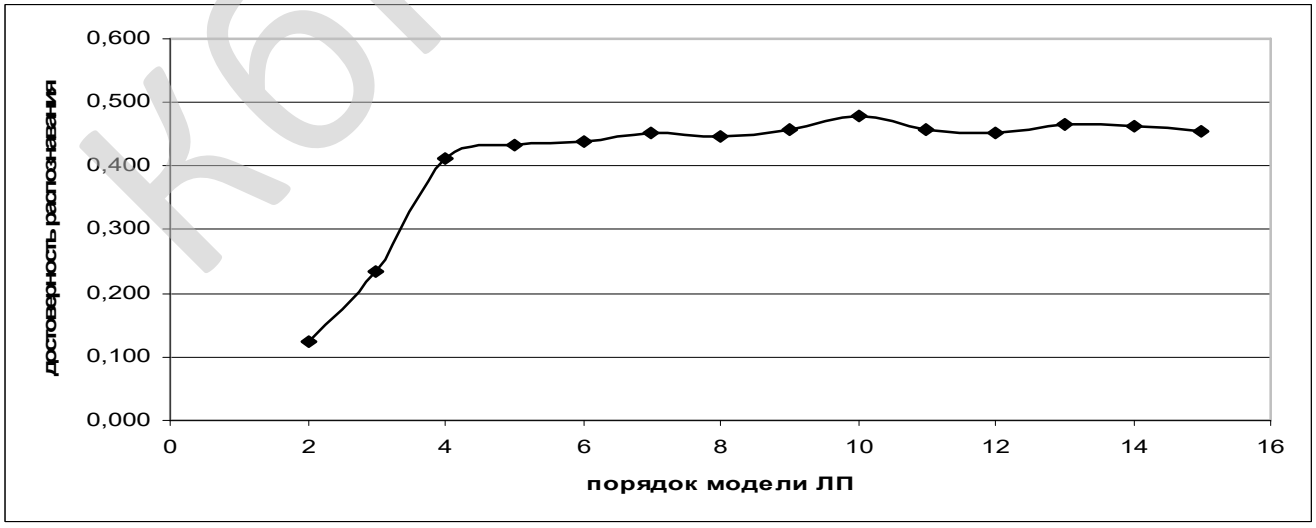

Рисунок 3.15 – Вплив порядку моделі на вірогідність розпізнавання

Результати досліджень погодяться із загальновідомими оцінками оптимальних параметрів моделі ЛП. Що підтверджує адекватність запропонованого критерію оцінки вірогідності розпізнавання мовних команд.

Оцінка якості сформованого словника еталонів

При використанні системи розпізнавання мовних команд в умовах підвищеної зашумленісті або на вузькополосних каналах зв'язку, навіть на словниках малих обсягів (до 50 слів) можливо велика кількість помилок. Для збільшення надійності запропоновано використовувати корекцію словника еталонів.

Таблиця 3.3 – Аналіз того, наскільки елементи відрізняються друг від друга

Еталони	Календар	Контакти	Наналашт	Офіс	Пам'ять	Режими	Зв'язок	Повідомле	Засоби	Теми	Годинник
Журнал	2,64	2,59	3,11	2,45	3,47	2,73	3,34	4,73	2,8	2,39	3,54
Календар		2,54	3	2,78	2,96	2,4	2,72	4,24	3,15	2,28	3,47
Контакти			2,77	2,87	3,32	3,09	3,17	4,35	3,1	2,52	3,49
Наналаштування				2,24	2,53	3,15	3,41	3,07	3,46	2,82	3,33
Офіс					2,36	2,71	2,59	3,48	3,31	2,46	3,2
Пам'ять						3,51	2,66	2,6	4	3,06	3,7
Режими							2,59	3,44	2,72	*1,95*	3,23
Зв'язок								4,29	3,01	2,4	2,98
Повідомлення									4,16	3,9	3,23
Засоби										2,4	3,2
Теми											2,88
Середнє	2,83	3,07	2,99	2,77	3,11	2,87	3,01	3,77	3,21	2,64	3,30

Після формування словника виробляється аналіз того, наскільки елементи відрізняються друг від друга (табл. 3.3). Підраховується середнє значення (у даному прикладі 3,04). Якщо деякі елементи словника занадто схожі один на одного (міра близькості менше порога, рівного 2), то пропонується замінити один з еталонів, наприклад, синонімом. Після цього виробляється повторний аналіз словника.

У даному прикладі (табл. 3.3), після заміни одного зі схожої пари слів «теми» або «режими», відсоток правильно розпізнаних команд збільшився на 9,8%.

У табл. 3.4 показані результати розпізнавання для чотирьох варіантів первинних ознак:

− LSP – лінійних спектральних корінь (пари)

− LPC – коефіцієнти лінійного пророкування

− PLP – коефіцієнти перцептивного пророкування

− MFCC – мел-кепстральні коефіцієнти

Таблиця 3.4 – Результати розпізнавання

Диктори	Час розрахунку, мс				Вірогідність				% помилок			
	LSP	LPC	PLP	MFCC	LSP	LPC	PLP	MFCC	LSP	LPC	PLP	MFCC
Чоловік1	9,05	3,05	13,09	13,02	1,91	1,30	1,49	1,13	0,00	4,76	2,38	2,38
Чоловік2	8,45	2,85	11,91	12,30	0,78	0,41	0,78	0,69	4,76	23,80	23,80	19,05
Жінка1	8,31	2,24	12,20	11,50	1,46	0,90	1,07	1,06	3,84	19,23	7,69	7,69
Жінка2	7,95	2,15	10,30	10,23	1,35	0,75	0,99	0,95	2,86	8,57	5,71	5,71
Середнє	8,44	2,57	11,88	11,76	1,38	0,84	1,08	0,96	2,87	14,09	9,90	8,71

Видно, що для ЛСК спостерігається мінімальний відсоток помилок 2,87% і максимальний ступінь вірогідності 1,38. При цьому час розрахунку порівнянний з іншими методами. Що дозволяє говорити про можливість успішного застосування даних ознак у більше складних системах розпізнавання мови.

3.2 Розробка структурної схеми

Для реалізації алгоритму розпізнавання мови, був обраний алгоритм MFCC.

Опис алгоритму MFCC

Існують безліч методів розпізнавання мови, у переважній більшості випадків вони засновані на методах статистичного аналізу й теорії ймовірностей (Hidden Markov Model, Gaussian Mixture Model і т.п.). Як відомо, компанія google надає безкоштовний сервіс по розпізнаванню коротких мовних повідомлень. На основі цього сервісу було навіть запропоноване розпізнавання мови за допомогою мікроконтролера. Однак, виникає питання: є чи можливість зробити свою систему розпізнавання мови, нехай навіть на досить обмеженому за розміром словнику, без використання «зовнішніх» сервісів, при цьому щоб вона працювала швидко й із прийнятною якістю?

Основна ідея

Отже, для розпізнавання мови будемо використовувати MFCC. Щоб не вдаватися в подробиці скажу, що відноситися до них варто лише як до деякого фільтра, на вході якого – фонограма, на виході – набір векторів (коефіцієнти), що ми й будемо розпізнавати як деяке слово або набір слів.

Справедливості заради варто відзначити, що існують безліч інших акустичних ознак, що використовуються для розпізнавання мови: Perceptual linear predictive (PLP), Linear prediction cepstral coefficient (LPCC), Linear frequency cepstral coefficients (LFCC).

Основна ідея полягає у використанні лінійного дискрімінантного аналізу для ідентифікації слова. Однак, він застосовний лише для векторів однакової розмірності. Т.к. слова можуть бути різної довжини, виникає питання: яким образом перетворити послідовність довільного числа MFCC-векторів у вектор фіксованої розмірності?

Можна надійти в такий спосіб: знаходити місця «згущення» розподілу цих векторів і в якості результуючого вектора брати конкатенацію векторів, що є центрами «згущень».

Такий конкатенований вектор будемо називати супервектором середніх, а самі центри – середніми значеннями. При цьому в якості «відправної точки» будемо використовувати супервектор середніх, отриманий на всіх MFCC-векторах всієї бази навчання.

Перетворивши в такий спосіб послідовність MFCC-векторів в один супервектор середніх фіксованої розмірності, ми можемо застосовувати різні методи класифікації.

Очевидний принциповий недолік такого підходу: не враховується динаміка розподілу MFCC-ознак за часом, отже, система апріорі не здатна розрізняти, приміром, слова «головриба» і «абирвалг», тому що загальний розподіл MFCC-векторів таких слів буде приблизно однаковим (відповідно, центри «згущень» будуть збігатися).

Опис алгоритму

Як базу навчання будемо використовувати безліч файлів, кожний з яких являє собою набір MFCC-векторів, отриманих з фонограми із записом того або іншого слова. При цьому файли із записом того самого слова повинні бути об'єднані в одну групу.

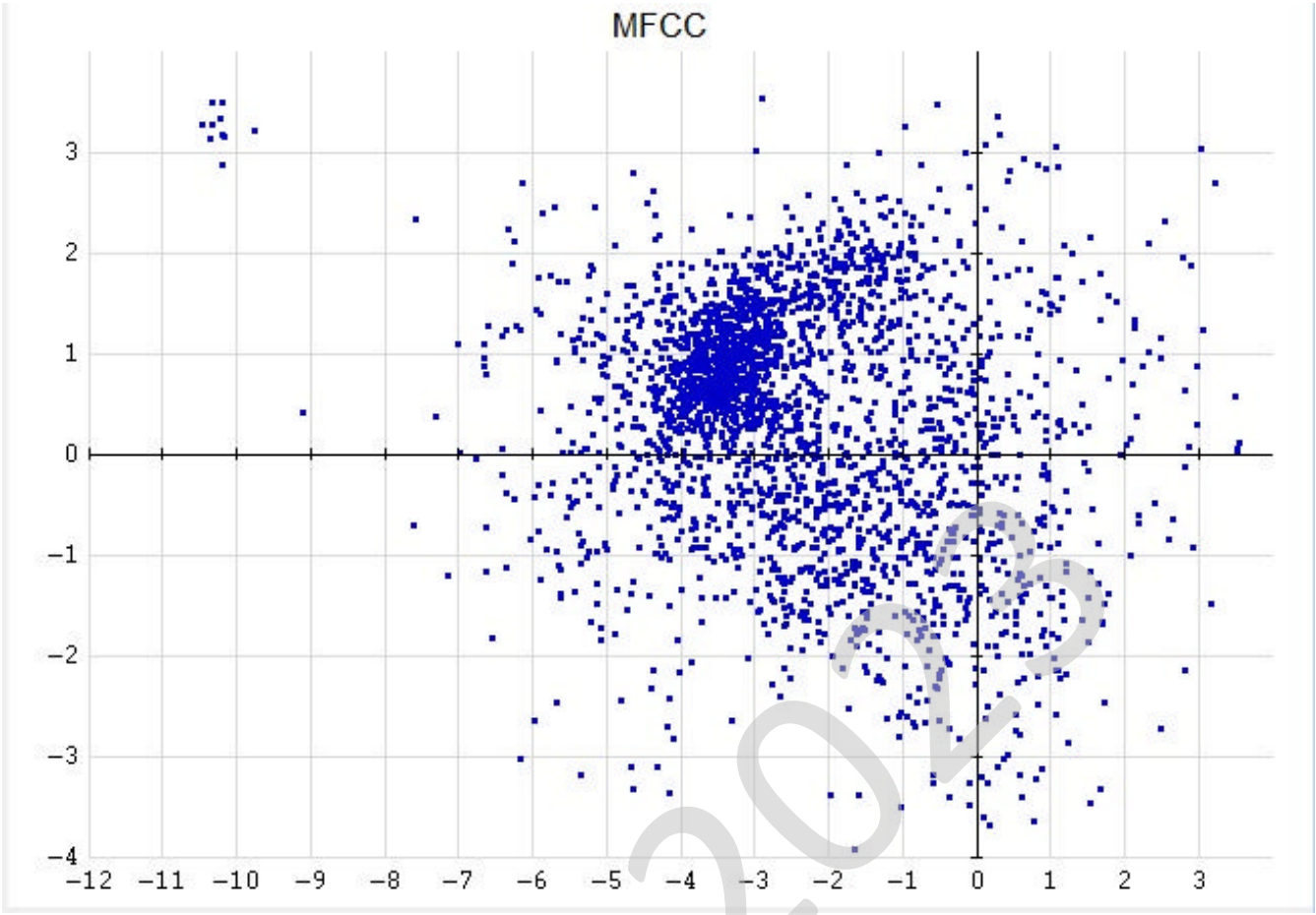

Рисунок 3.16 – Розподіл перших двох компонентів MFCC-векторів всієї бази навчання

Алгоритм складається з наступних етапів:

1. Знаходимо супервектор середніх для всієї бази навчання за допомогою алгоритму K-середніх.

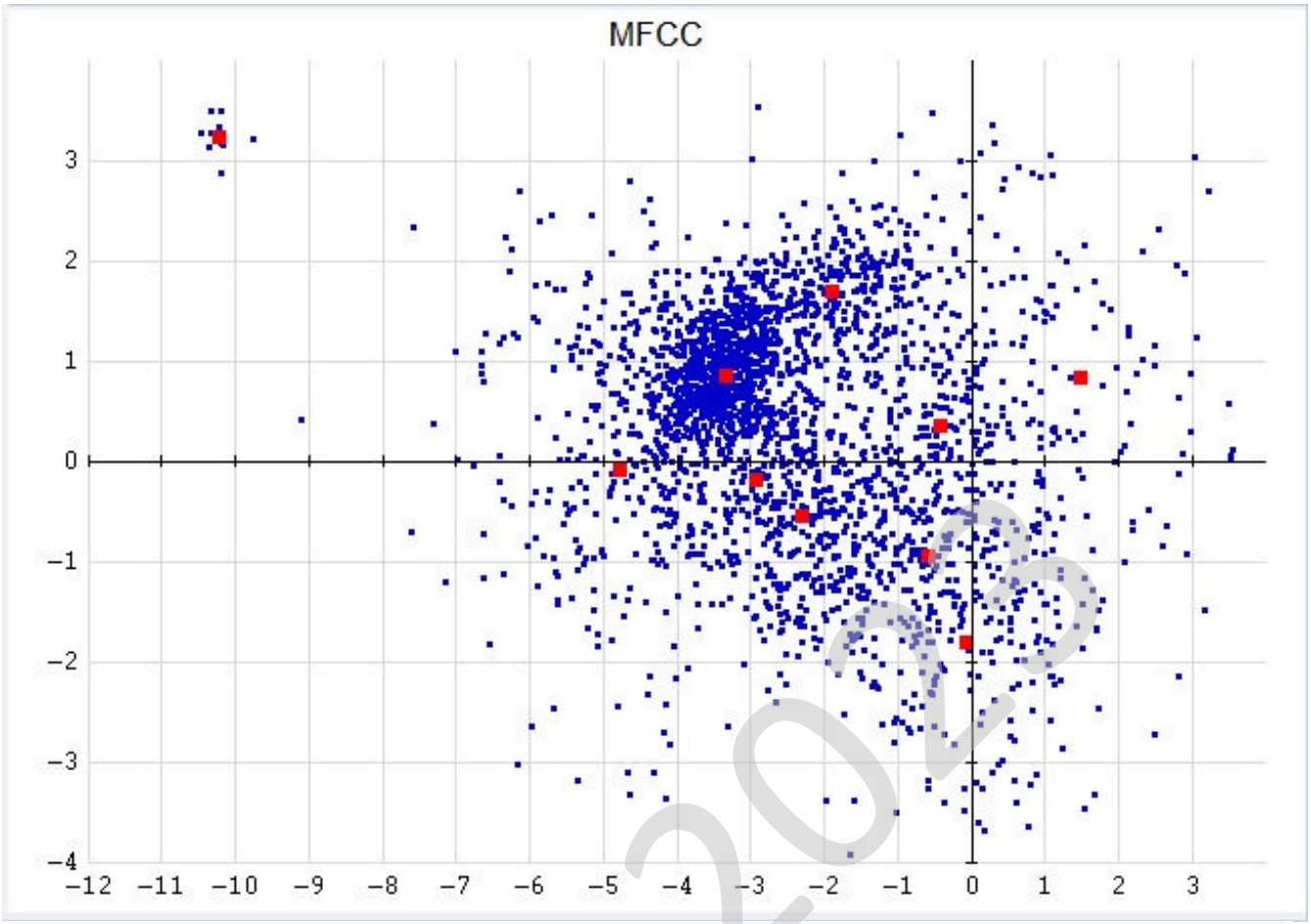

Рисунок 3.17 – Приклад роботи алгоритму K-середніх для K=10, де більші червоні квадрати і є шукані середні значення

2. Для кожного файлу бази знаходимо власні середні значення за формулою:

$$Mk = a * Mk0 + (1 – a) * Mk', k = 1:K$$

де Mk0 – середнє значення, знайдене в п.1, Mk' – середнє значення, отримане в результаті застосування однієї ітерації алгоритму K-середніх для MFCC-векторів файлу з використанням як початкове значення Mk0,

$$a = R/(R + Nk),$$

де R – коефіцієнт «чутливості», Nk – число MFCC-векторів, що відповідають середньому значенню Mk'.

3. Знайдені в такий спосіб середні значення будемо називати адапритованими середніми значеннями.

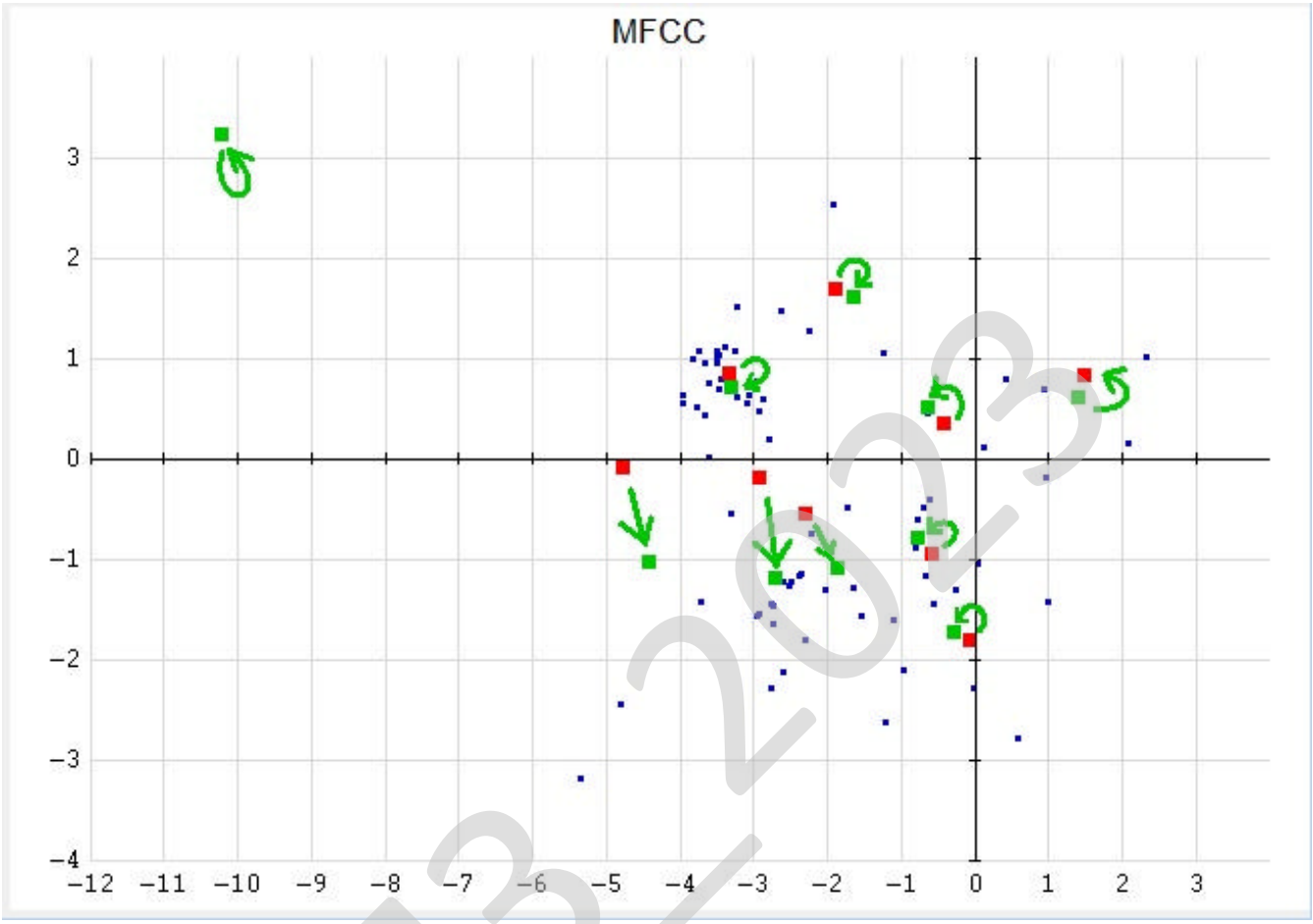

Рисунок 3.18 – Приклад адаптованих середніх значень для файлу

4. Маючи тепер замість вихідних фонограм адаптовані супервектора середніх, проводимо LDA для N класів (кожний клас відповідає одному слову).

5. У результаті ми повинні одержати матрицю, що складається з векторів нового базису, при проекції на який вихідні адаптовані супервектора середніх повинні досить добре розділятися.

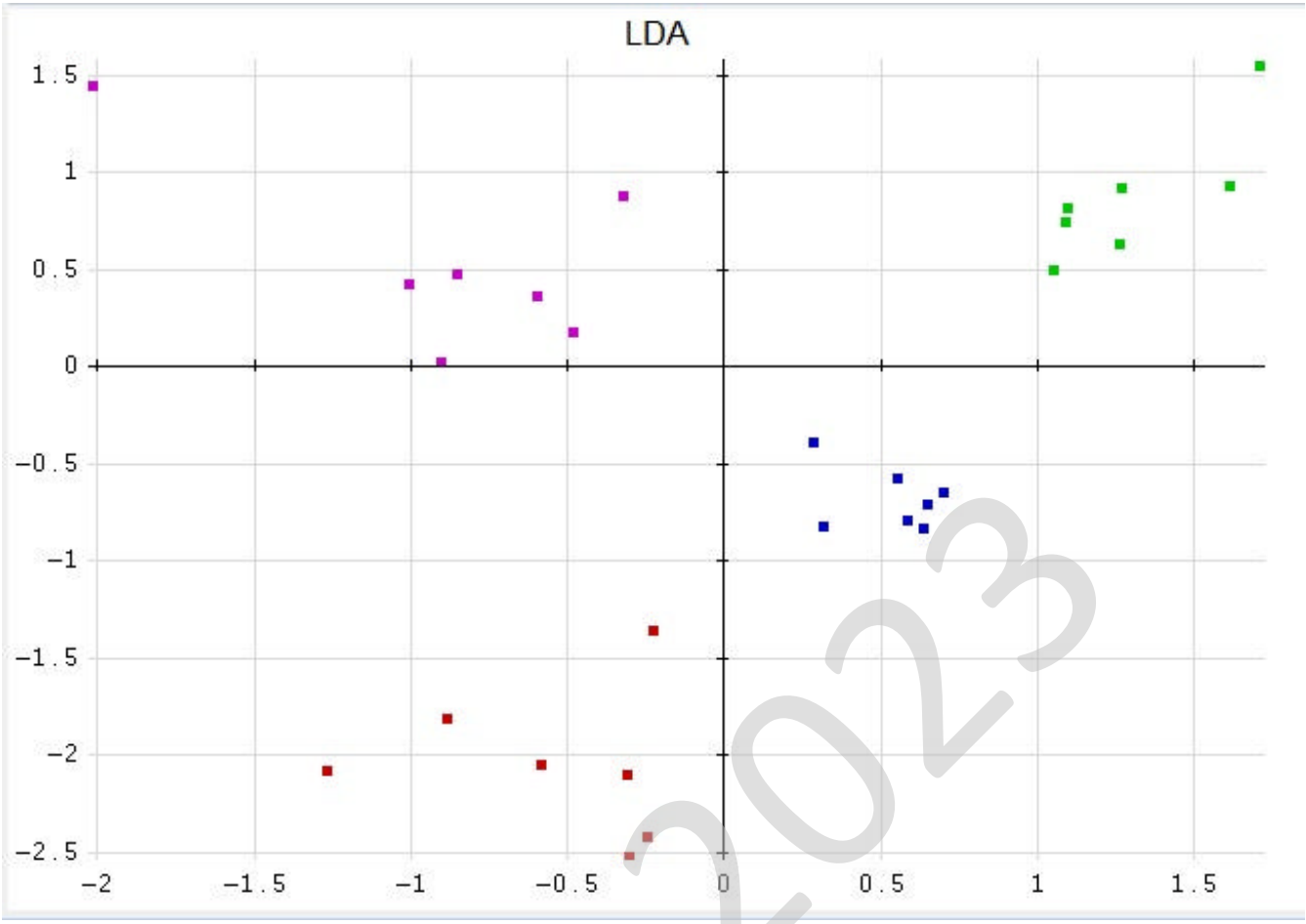

Рисунок 3.19 – Приклад для N=4

6. Проектуємо всі адаптовані супервектора середніх на новий базис і знаходимо середні значення й СКВ (середнє квадратичне відхилення) проекцій для кожного класу.

7. Для визначення приналежності тестової фонограми тому або іншому класу (тобто розпізнавання), виконуємо для неї пп. 2 і 4, далі знаходимо відстані отриманої проекції до середніх значень всіх класів (можна додатково нормувати їх на відповідне СКВ). Мінімальна відстань і буде відповідати класу, до якого належить тестова фонограма.

Реалізація

Створення власної системи розпізнавання слів складається з наступних етапів:

1. Запис фонограм для навчання й тестування.

Для запису можна скористатися будь-якою програмою, що вміє записувати звук і зберігати його у форматі WAVE. Я рекомендую використовувати безкоштовну програму Audacity.

Розроблена система не вміє виділяти мовні сегменти, тому при записі потрібно намагатися, щоб у фонограмі була присутня тільки мова. Чим якісніше використовується мікрофон, тим якіснішою виходить система. Записувати необхідно в моно-режимі із частотою дискретизації 16000.

2. Побудова MFCC-векторів.

Для побудови MFCC-векторів можна використовувати безкоштовну бібліотеку SPro 5.0. Я взяв на себе відповідальність, небагато перебрав цю бібліотеку, виправив парочку помилок і зробив складання програми sfbcep.exe під windows (див. папку ../ spro-5.0). 32-розрядна версія цієї програми лежить у папці ../tools. Для побудови MFCC-векторів я використовував наступні параметри:

sfbcep.exe -format=wave – sample-rate=16000 -mel – freq-min=0 – freq-max=8000 – fft-length=256 -length=16.0 -shift=10.0 – num-ceps=13 [вхідний WAVE-Файл] [вихідний файл із MFCC-векторами]

Навчання й тестування системи

Для навчання й тестування системи я написав програму wrsystem мовою visual C++. Реалізація алгоритму LDA була запозичена з бібліотеки ALGLIB.

Програма wrsystem має два режими роботи: навчання (у випадку наявності параметра -learn) і тестування. Ця програма приймає на вхід три основних параметри:

– Шлях до файлу з описом бази навчання (тестування) (параметр -base). Приклад файлу з описом бази лежить у папці ../base, також опис формату можна подивитися, запустивши програму з параметром -help.

– Шлях до бінарного файлу, що зберігає результат навчання системи (параметр -system). У режимі навчання цей файл створюється, у режимі тестування – зчитується.

– Шлях до файлу, у який записуються результати тестування системи на зазначеній базі: матриця переплутування й значення WER (Word Error Rate) (параметр -test_results**).**

Результати експериментів

Як експеримент я створив систему, що вміє розпізнавати 14 слів, записаних моїм голосом. Для навчання системи я записав кожне слово 4-5 разів, а для тестування – 7 разів. Разом база навчання містить 63 файлу, а база тестування – 98. Використовувалися наступні параметри при навчанні:

− Кількість середніх значень: 10.

− Коефіцієнт «чутливості» при адаптації: 20.

− Розмірність проекції: 20.

− Використання нормалізації на СКВ: відсутній.

Результат тестування на базі навчання показав рівень помилки розпізнавання слів (WER) 1,6%, а на базі тестування 5,1%.

На що варто звернути увагу

Для того, щоб будь-яка система (включаючи описану тут) могла якісно розпізнавати мову будь-якої людини, необхідно мати величезну базу навчання із записом всіх слів, вимовлених різними людьми в різному емоційному стані з використанням різних записуючих пристроїв (телефон, мікрофон, що підслухує пристрій і т.п.). Тобто система, що ви навчите, використовуючи тільки свій голос і тільки вашу домашню гарнітуру, напевно не буде працювати для ваших знайомих і навіть для вас, якщо ви будете використовувати який-небудь інший мікрофон.

Структурна схема розробленої системи зображена на рисунку 3.20. На ній показано структуру системи дистанційного голосового керування робототехнічним комплексом.

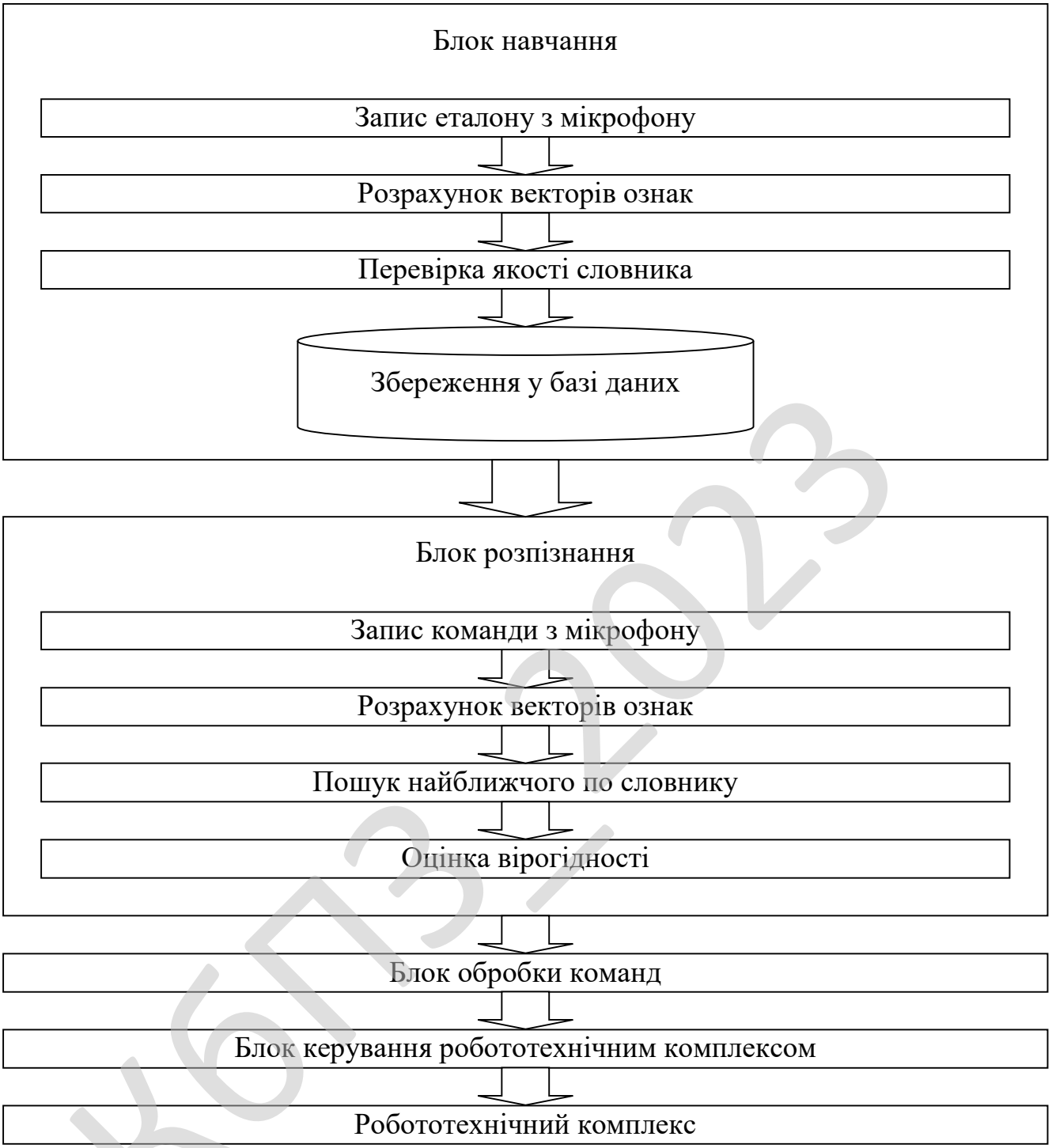

Рисунок 3.20 – Структурна схема системи

3.3 Розробка функціональної схеми

Представлено алгоритмічну модель системи, на підставі якої в будь-якому сучасному середовищі розробки можлива побудова програмного комплексу, що використовує процедуру розпізнавання команд. Крім того, окремі модулі системи готові до реалізації на базі програмувальних апаратних засобів (DSP-процесори, Плиси й т.д.) з можливістю розпаралелювання обчислювальних операцій.

У роботі проведене тестування моделі розпізнавання мовних команд. Виконано порівняльний аналіз ЛСК і інших розповсюджених методів одержання первинних ознак мовних сигналів. Порівняння проводилося на тих самих тестових зразках.

Як еталонна база використовувалися 42 командні слова, надиктованих чотирма дикторами. На вхід подавалися сигнали від цих же дикторів, по одному варіанті проголошення кожної команди. Оцінювався відсоток помилок і середня вірогідність розпізнавання. Якщо була допущена помилка, то поточна команда не брала участь у підрахунку середнього. Також був підрахований середній час розрахунку набору первинних ознак для одного командного слова.

Функціональна схема розробленої системи зображена на рисунку 3.21. Розглянемо більш детально кожний з блоків розробленої програми.

З рисунку видно, що розроблена система складається з наступних частин:

1. Головне вікно інтерфейсу користувача системи дистанційного голосового керування робототехнічним комплексом. На базі модулів програмного комплексу побудована досвідчена система розпізнавання мовних команд. Система виконує всі операції, починаючи із запису вхідного сигналу з мікрофона й закінчуючи видачею розпізнаної команди у вигляді текстового повідомлення на екрані ПК.

2. Блок диктування тексту. Коли користувач вимовляє слова в мікрофон, програма розпізнавання тексту перетворить вимовлені слова в текст, що з'являється на екрані.

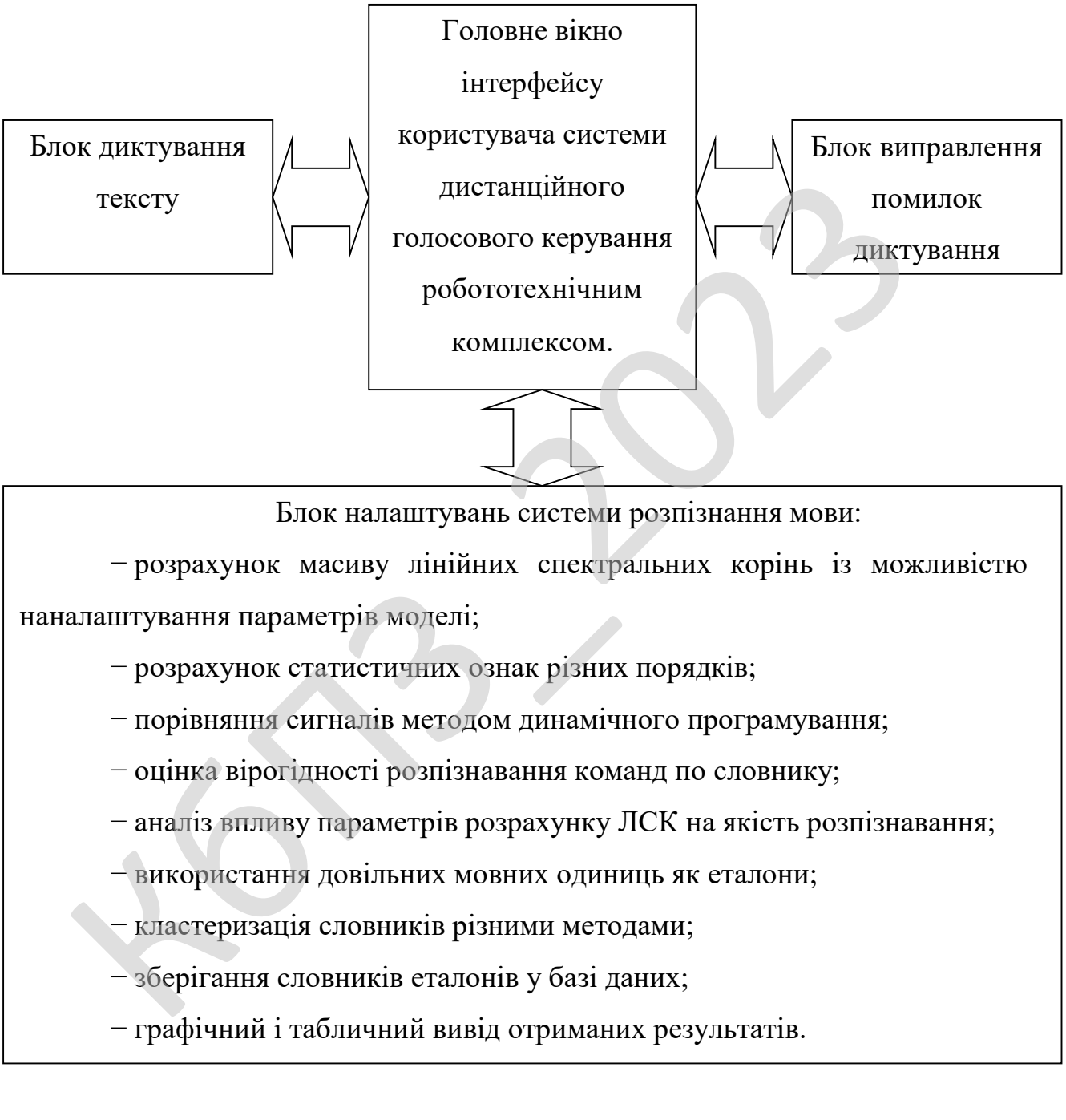

Рисунок 3.21 – Функціональна схема системи

3. Блок виправлення помилок диктування. Існує кілька способів виправлення помилок, допущених під час диктування. Для виправлення останньої вимовленої фрази необхідно сказати «Виправити це». Щоб виправити одне слово, вимовте «виправити», а потім це слово. Якщо слово повторюється кілька разів, то всі його екземпляри будуть виділені й можна вибрати з них той, котрий потрібно виправити. Можна також додати слова, які часто невірно сприймаються або не розпізнаються програмою розпізнавання мови.

4. Блок налаштувань системи розпізнання мови:

− розрахунок масиву лінійних спектральних корінь із можливістю наналаштування параметрів моделі;

− розрахунок статистичних ознак різних порядків;

− порівняння сигналів методом динамічного програмування;

− оцінка вірогідності розпізнавання команд по словнику;

− аналіз впливу параметрів розрахунку ЛСК на якість розпізнавання;

− використання довільних мовних одиниць як еталони;

− кластеризація словників різними методами;

− зберігання словників еталонів у базі даних;

− графічний і табличний вивід отриманих результатів.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.22. Першим процесом, який завантажується у системі, є процес виведення головного вікна програми, який взаємодіє з наступними процесами:

– Процес ведення журналу подій.

– Процес запису звуку з мікрофону.

– Процес роботи з базою даних.

Процес запису звуку з мікрофону взаємодіє з наступними процесами:

– Процес виведення часової діаграми звуку.

– Процес розпізнавання звуку.

Процес розпізнавання звуку взаємодіє з наступними процесами:

– Процес обробки помилок розпізнавання.

– Процес визначення голосової команди.

Рисунок 3.22 – Діаграма взаємодії процесів

Процес визначення голосової команди взаємодіє з наступними процесами:

– Процес розрахунку векторів ознак.

– Процес пошуку найближчого по словнику шаблону.

– Процес виконання команди робототехнічним комплексом.

Процес виконання команди робототехнічним комплексом взаємодіє з наступним процесом:

– Процес введення результатів виконання програми.

– Процес виведення розпізнаної команди у текстовому форматі.

Процес роботи з базою даних взаємодіє з наступними процесами:

– Процес навчання системи по еталоним зразкам, який взаємодіє з процесом виведення результатів навчання.

– Процес перегляду бази даних, який взаємодіє з процесом редагування бази даних.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми, блок-схема якого зображена на рисунку 4.1.

Після виведення основного вікна програми, користувачеві пропонується вибір, чи потрібно здійснювати навчання системи. Якщо «Так» відбувається навчання системи по еталонним зразкам, після чого виводиться звіт результатів навчання.

На наступному кроці перевіряється чи потрібно ввімкнути мікрофон, якщо цього не відбувається, програма завершує роботу.

Якщо мікрофон ввімкнено, при поданні голосових команд після зчитування звуку із мікрофону на екран виводиться часова діаграма звуку.

Далі виконується підпрограма розпізнавання голосових команд.

Якщо робота підпрограми є успішною і голосову команду розпізнано, виконуються наступні операції:

– Виведення розпізнаної команди у вигляді тексту на екран.

– Виконання команди управління робототехнічним комплексом на основі розпізнаної команди.

– Виведення звіту про виконання команди.

У випадку, якщо голосову команду не вдалося розпізнати, виводиться повідомлення про те, що команда не розпізнана.

Крім навчання та виконання команд робототехнічним комплексом, із основної програми можливо також отримати доступ до бази даних голосових команд.

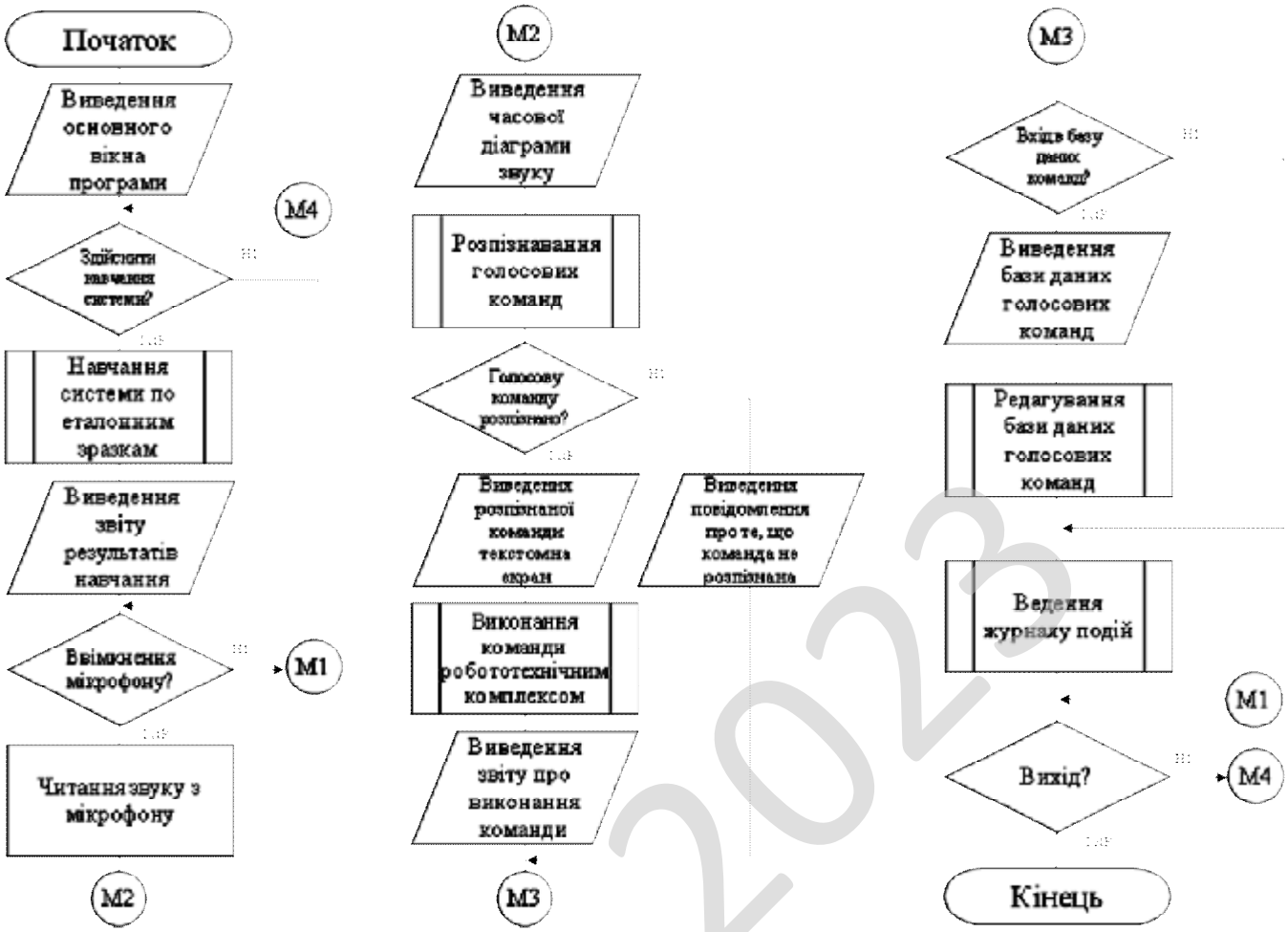

Рисунок 4.1 – Блок-схема основної програми

При поданні запиту на доступ до бази даних виконуються наступні операції:

– Виведення бази даних голосових команд.

– Редагування бази даних голосових команд користувачем.

Під час роботи програми відбувається ведення журналу подій.

Розглянемо підпрограму навчання системи розпізнавання голосових команд по еталонним зразкам. Її блок-схема наведена на рисунку 4.2.

Робота підпрограми складається із наступних кроків:

Крок 1. Читання з бази даних еталонних зразків голосових команд.

Крок 2. Формування набору вхідних та вихідних векторів для навчання у кількості K.

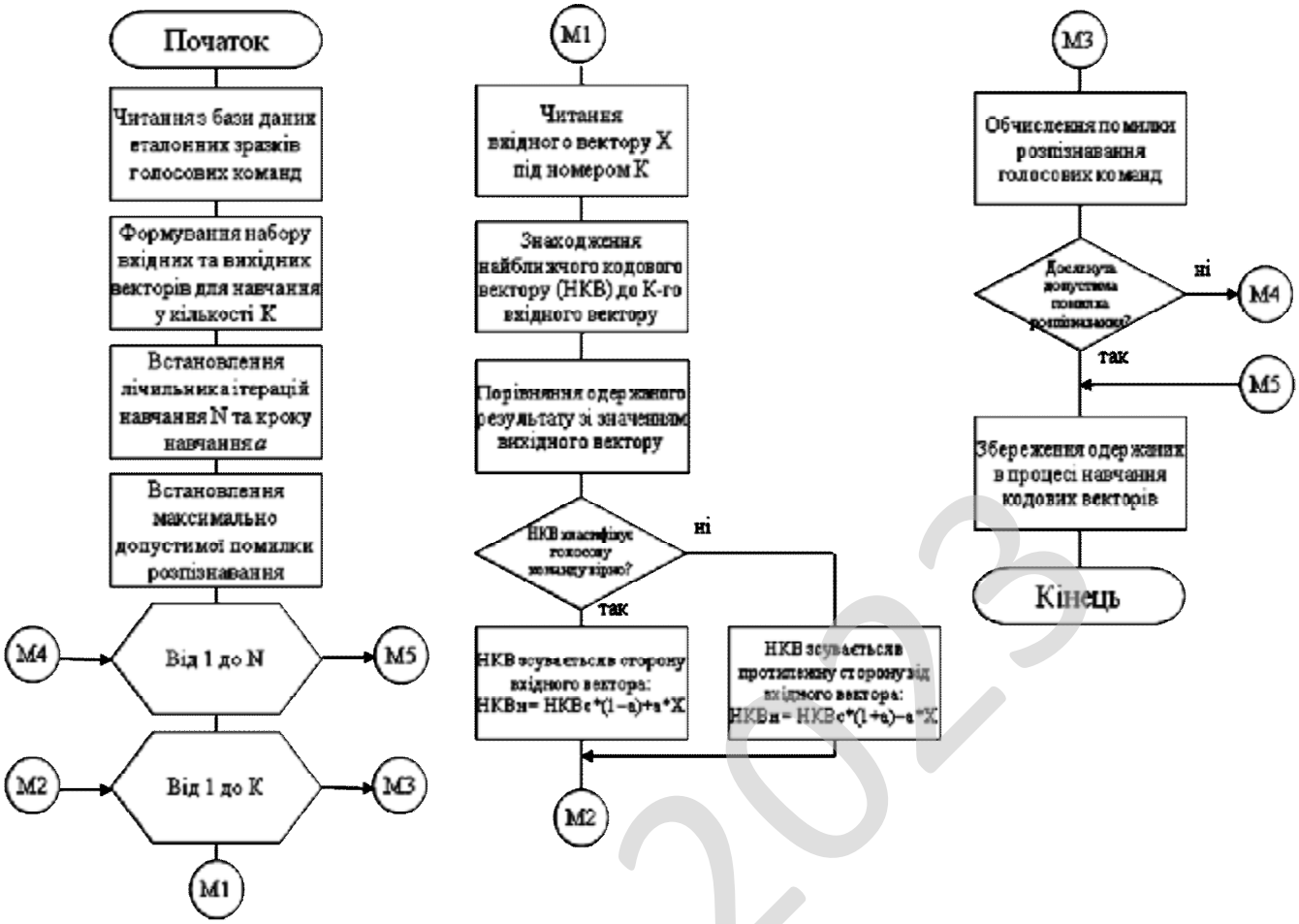

Рисунок 4.2 – Блок-схема підпрограми навчання системи розпізнавання голосових команд по еталонним зразкам

Крок 3. Встановлення лічильника ітерацій навчання N та кроку навчання a.

Крок 4. Встановлення максимально допустимої помилки розпізнавання.

Крок 5. Цикл 1 від 1 до N.

Крок 6. Цикл 2 від 1 до K.

Крок 7. Обчислення помилки розпізнавання команд.

Крок 8. Читання вхідного вектору X під номером K.

Крок 9. Знаходження найближчого кодового вектора (НКВ) до K-го вхідного вектора.

Крок 10. Порівняння одержаного результату зі значенням вихідного

вектора.

Крок 11. Якщо НКВ класифікує голосову команду вірно, крок 12, інакше – крок 13.

Крок 12. НКВ зсувається в сторону вхідного вектора:

НКВн = НКВс*(1–а)+а*Х.

Крок 13. НКВ зсувається в сторону, протилежну до вхідного вектора:

НКВн = НКВс*(1+а)–а*Х.

Крок 14. Оновити значення в циклі 2, перейти до кроку 6.

Крок 15. Якщо досягнута допустима помилка розпізнавання, перейти до кроку 16, в протилежному випадку – оновити значення в циклі 1, перейти до кроку 5.

Крок 16. Зберегти одержані в процесі навчання кодові вектори.

Крок 17. Завершення роботи підпрограми.

Навчання системи в програмі відбувається за допомогою наступних методів:

– KMeans: Знаходження середніх векторів на всій базі навчання.

– Adaptation: Адаптація середніх векторів для кожного файлу з бази навчання.

– CreateLDAMatrix: Побудова LDA-матриці для адаптованих середніх векторів.

– MakeProjection: Знаходження проекцій адаптованих середніх векторів на площину LDA-матриці.

– CreateCompareSystem: Знаходження середніх значень і СКО знайдених проекцій усередині кожного класу.

Порівняння тестованого та кодових векторів відбувається за допомогою методу CreateCompareSystem, основна частина коду якого наведена нижче:

```
ResultStatus WordRecognizer::CreateCompareSystem(
   const IVectorsSet  &_vectorsBase,
   float*                  _meanBase,
   float*                  _sigmaBase,
   const float             &_minSigma)
```

```
{
    int meansNum = _vectorsBase.GetSize();
    int vectorsSize = _vectorsBase.GetDim();
    ResultStatus result = RS_SUCCESS;

    float d = 0;
    float x = 0;
    float sigma = 0;
    float minD = FLT_MAX;
    int vectorsNum = 0;
    int i, j, k;
    const float* pVector = NULL;
    float* pMean = NULL;

    memset(_meanBase, 0, sizeof(float) * meansNum * vectorsSize);
    memset(_sigmaBase, 0, sizeof(float) * meansNum);

    pMean = _meanBase;
    for (i = 0; i < meansNum; i++, pMean += vectorsSize)
    {
        _vectorsBase.GetVectors(i, vectorsNum, pVector);
        sigma = 0;
        for (j = 0; j < vectorsNum; j++, pVector += vectorsSize)
        {
            for (k = 0; k < vectorsSize; k++)
            {
                x = pVector[k];
                pMean[k] += x;
                sigma += x * x;
            }
        }
        x = 1.f/(float)vectorsNum;
        d = 0;
        for (k = 0; k < vectorsSize; k++)
        {
            pMean[k] *= x;
            d += pMean[k] * pMean[k];
        }
        _sigmaBase[i] = sqrt(fabs(sigma * x - d));
        if (_sigmaBase[i] < _minSigma)
        {
            result = RS_MIN_SIGMA_USED;
```

```
                _sigmaBase[i] = _minSigma;
            }
        }
        return result;
    }
```

Розглянемо роботу підпрограми розпізнавання голосових команд. Її блок-схема наведена на рисунку 4.3.

Загальний порядок виконання цієї програми є наступним:

– Після прочитання голосової команди, поданої з мікрофону відбувається попередня обробка звуку.

– Виконується дискретне перетворення Фур’є отриманого сигналу.

– Отримується вектор ознак.

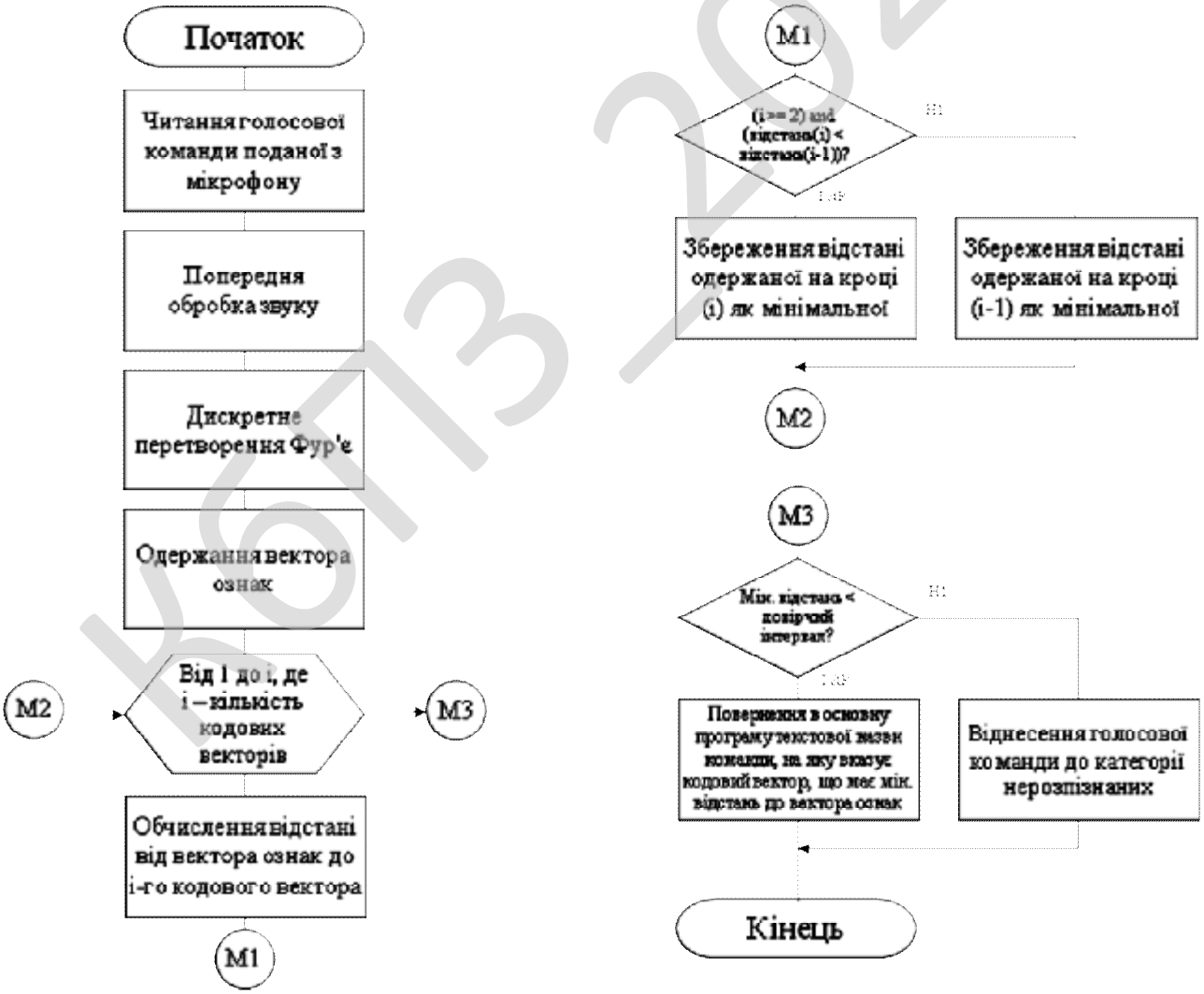

Рисунок 4.3 – Блок-схема підпрограми розпізнавання голосових команд

– В циклі обчислюється мінімальна відстань від одного із і кодових векторів до вектора ознак.

– Якщо мінімальна відстань менша довірчого інтервалу, в основну програму повертається текстова назва команди, на яку вказує кодовий вектор (вектор, що має мінімальну відстань до вектора ознак). В протилежному випадку команда відноситься до категорії нерозпізнаних.

Запис аналогового сигналу із зовнішнього джерела здійснюється за допомогою мікрофона. У результаті такої операції ми одержимо набір значень, які відповідають зміні амплітуди звуку у часі. Такий принцип кодування називається імпульсно-кодовою модуляцією PCM (Pulse-code modulation). «Сирі» дані, отримані з аудіо-потоку, поки ще не підходять для поставлених цілей. Першою справою потрібно перетворити біти в набір осмислених значень – амплітуд сигналу. У якості вхідних даних використовується нестиснутий 16-бітний знаковий (PCM-signed) wav-файл із частотою дискретизації 16 кГц.

Підпрограма підготовки аудіоданих до використання:

```
double[] readAmplitudeValues(bool isBigEndian)
{
 int MSB, LSB; // старший і молодший байти
 byte[] buffer = ReadDataFromExternalSource(); // читаємо дані звідки-небудь
 double[] data = new double[buffer.length / 2];

 for (int i = 0; i < buffer.length; i += 2)
 {
     if(isBigEndian) // задаємо порядок байтів у вхідному сигналі
     {
             // першим байтом буде MSB
             MSB = buffer[2 * i];
             // другим байтом буде LSB
             LSB = buffer[2 * i + 1];
     }
     else
     {
             // навпаки
             LSB = buffer[2 * i];
             MSB = buffer[2 * i + 1];
```

```
            }
            // склеюємо два байти, щоб одержати 16-бітне дійсне число
            // всі значення діляться на максимально можливе – 2^15
            data[i] = ((MSB << 8) || LSB) / 32768;
        }
        return data;
    }
```

При обробці звуку отримані значення амплітуд можуть не збігатися навіть для двох однакових записів через зовнішній шум, різні гучності вхідного сигналу й інших факторів. Для приведення звуків до «загального знаменника» використовується нормалізація. Ідея пікової нормалізації проста: розділити всі значення амплітуд на максимальну (у рамках даного звукового файлу). У такий спосіб зрівнюються зразки мови, записані з різною гучністю, уклавши все в шкалу від -1 до 1. Важливо, що після такої трансформації будь-який звук повністю заповнює заданий проміжок.

Нормалізація – найпростіший і ефективний алгоритм попередньої обробки звуку. Існують також маса інших: «відрізаючі» частоти вище або нижче заданої, згладжуючі та ін.

Розглянемо реалізацію на основі алгоритму «розділяй та пануй». Навіть при роботі зі звуком з мінімально достатньою частотою дискретизації (16 кГц) розмір унікальних характеристик для секундного зразка звуку просто величезний – 16000 значень амплітуд. Робити скільки-небудь складні операції над такими обсягами даних не представляється можливим. Крім того, не зовсім зрозуміло, як порівнювати об'єкти з різною кількістю унікальних рис.

Для початку знизимо обчислювальну складність задачі, розбивши її на менші по складності підзадачі. Цим ходом вирішуємо одрузу дві задачі, адже встановивши фіксований розмір підзадачі й усереднивши результати обчислень по всіх задачах, одержимо наперед задану кількість ознак для класифікації.

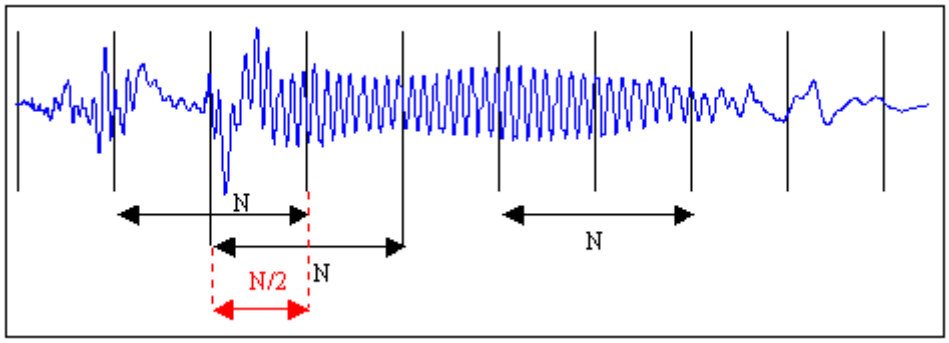

Рисунок 4.4 – «Розрізання» звукового сигналу на кадри довжини N

На рисунку 4.4 зображено «розрізання» звукового сигналу на кадри довжини N з половинним перекриттям. Необхідність у перекритті викликана перекручуванням звуку у випадку, якщо кадри були розташовані поруч. Хоча на практиці цим прийомом часто нехтують для економії обчислювальних ресурсів. Виберемо довжину кадру рівною 128 мс, як компроміс між точністю (довгі кадри) і швидкістю (короткі кадри). Залишок мови, що не займає повний кадр, можна заповнити нулями до бажаного розміру або просто відкинути.

Для усунення небажаних ефектів при подальшій обробці кадрів, помножимо кожний елемент кадру на особливу вагову функцію («вікно»). Результатом стане виділення центральної частини кадру й плавне загасання амплітуд на його краях. Це необхідно для досягнення кращих результатів при здійснені перетворення Фур'є, перетворення орієнтоване на нескінченно повторюваний сигнал. Відповідно, наш кадр повинен стикуватися сам із собою і як можна більш плавно. Вікон існує велика кількість, будемо використовувати вікно Хеммінга.

$$w(n) = 0.53836 - 0.46164 \cos\left(\frac{2\pi n}{N-1}\right),$$

де n – порядковий номер елемента в кадрі, для якого обчислюється нове значення амплітуди;

N – як і раніше, довжина кадру (кількість значень сигналу, вимірюваних за період).

Наступним кроком буде одержання короткочасної спектрограми кожного кадру окремо. Для цих цілей використовуємо дискретне перетворення Фур'є.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1,$$

де N – як і раніше, довжина кадру (кількість значень сигналу, обмірюваних за період);

x_n – амплітуда n-го сигналу;

X_k – N комплексних амплітуд синусоїдальних сигналів, що складають вихідний сигнал.

Крім цього, піднесемо кожне значення X_k у квадрат для подальшого логарифмування.

На сьогоднішній день найбільш успішними є системи розпізнавання голосу, що використовують знання про пристрій слухового апарата. Якщо говорити коротко, то вухо інтерпретує висоту тону звуків не лінійно, а в логарифмічному масштабі. Дотепер всі операції пророблялися над «герцами», тепер перейдемо до «мелів». Наочно представити залежність допоможе рисунок 4.5.

Як видно, мел-шкала поводиться майже лінійно до 1000 Гц, а після проявляє логарифмічну природу. Перехід до нової шкали описується нескладною залежністю.

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) = 1127 \log_e\left(1 + \frac{f}{700}\right),$$

де m – частота в мелах;

f – частота в герцах.

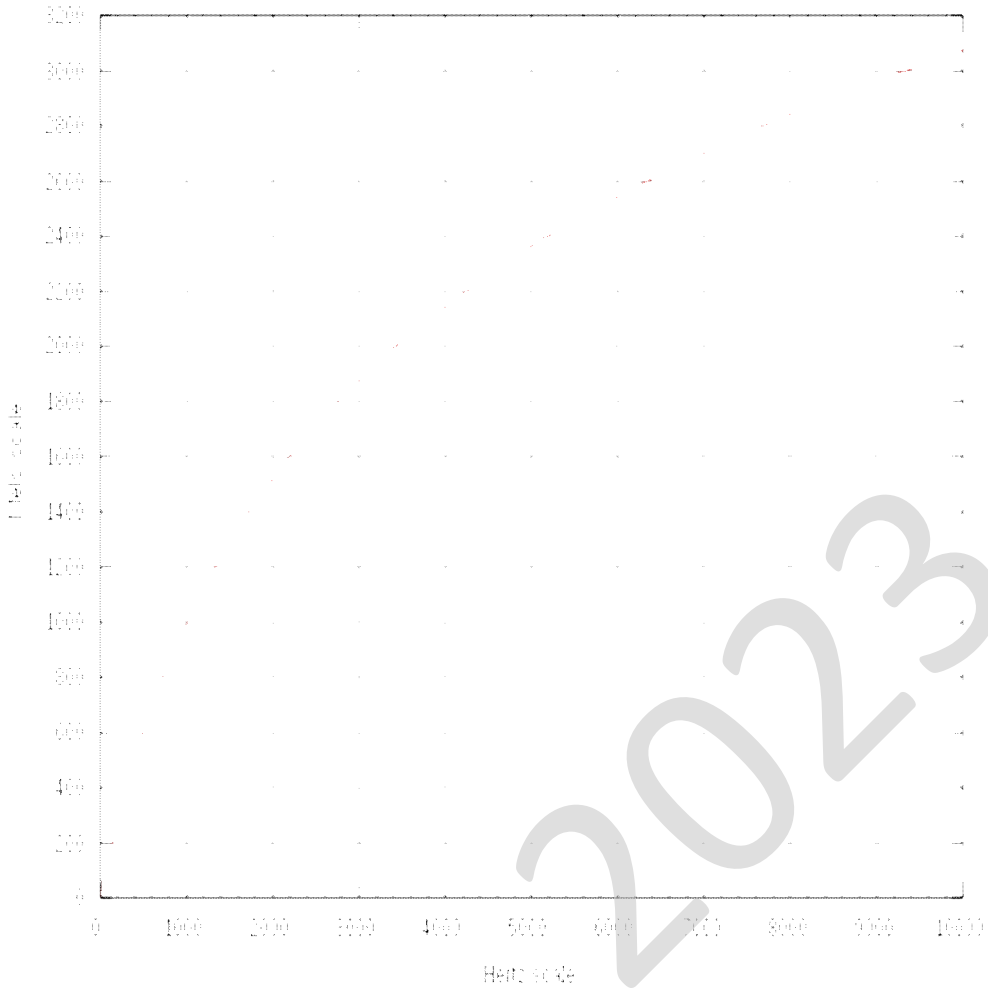

Рисунок 4.5 – Графік залежності між герц-шкалою та мел-шкалою

Вектор ознак буде складатися з тих самих мел-спектральних коефіцієнтів. Обчислюємо їх по наступній формулі:

$$\tilde{c_n} = \sum_{k=1}^{K} (\log \tilde{S}_k) \left[n \left(k - \frac{1}{2} \right) \frac{\pi}{K} \right],$$

де c_n – мел-спектральний коефіцієнт під номером n;

S_k – амплітуда k-го значення в кадрі в мелах;

K – наперед задана кількість мел-спектральних коефіцієнтів;

$n \in [1, K]$.

Останньою стадією є класифікація мовної команди. Класифікація здійснюється обчисленням міри схожості пробних даних і вже відомих. Міра

схожості виражається відстанню від вектора ознак пробного сигналу до вектора ознак уже класифікованого. Використаємо наступну формулу:

$$d_1(\mathbf{p},\mathbf{q}) = \|\mathbf{p}-\mathbf{q}\|_1 = \sum_{i=1}^{n} |p_i - q_i|,$$

Таке рішення більше підходить для векторів дискретної природи, на відміну від відстані Евкліда.

Знаходження вектора ознак для одного зразка зробити не важко: такий вектор представляється як середнє арифметичне векторів, що характеризують окремі кадри мови. Для підвищення точності розпізнавання просто необхідно усереднити результати не тільки між кадрами, але й ураховувати показники декількох мовних зразків. Маючи кілька записів голосу, розумно не усереднювати показники до одного вектора, а провести кластеризацію, наприклад за допомогою методу k-середніх.

Процес розпізнавання голосових команд можна побудувати у такий спосіб:

1. Збираємо декілька тренувальних зразків мови, чим більше – тим краще.

2. Знаходимо для кожного з них характеристичний вектор ознак.

3. Для зразків з відомим значенням команди проводимо кластеризацію з одним центром (усереднення) або декількома. Прийнятні результати починаються вже з використанням 4-х центрів для кожної команди.

4. У режимі впізнавання знаходимо відстань від пробного вектора до вивчених під час тренування центрів кластерів. До якого кластера пробна мова виявиться ближче – до такої команди й відносимо зразок.

5. Можна експериментально встановити навіть деякий довірчий інтервал – максимальна відстань, на якій може перебувати пробний зразок від центра кластера. У випадку перевищення цього значення – класифікувати зразок як невідомий.

Для визначення ймовірності правильності ідентифікації голосових команд використовується метод GetProbability, код якого наводиться нижче:

```
void WordRecognizer::GetProbability(
   const int     &_vectorsSize,
   const int     &_meansNum,
   const float*  _meanBase,
   const float*  _sigmaBase,
   const float*  _testVector,
   bool          _useSigma,
   float*             _result
   )
{
   int i, k;
   float d = 0;
   float x = 0;
   float sum = 0;
   float minD = 1.0e-10f;
   const float* pMean = _meanBase;
   for (i = 0; i < _meansNum; i++, pMean += _vectorsSize)
   {
        d = 0;
        for (k = 0; k < _vectorsSize; k++)
        {
             x = pMean[k] - _testVector[k];
             d += x * x;
        }
        if (d < minD) d = minD;
        _result[i] = (_useSigma) ? _sigmaBase[i] / sqrt(d) : 1.f / sqrt(d);
        sum += _result[i];
   }
   for (i = 0; i < _meansNum; i++)
        _result[i] /= sum;
}
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм REDOC III, який оперує з 80-бітовим блоком. Довжина ключа може змінюватися й досягати 2560 байт (204800 біт). Алгоритм складається тільки з операцій XOR над байтами ключа й відкритого тексту, перестановки й підстановки не використовуються.

1. Створюють таблицю ключів з 256 10-байтових ключів, використовуючи секретний ключ.

2. Створюють два 10-байтових блоки масок M1 і M2. M1 являє собою результат операції XOR перших 128 10-байтових ключів, а M2 – результат операції XOR других 128 10-байтовых ключів.

3. Для шифрування 10-байтового блоку:

a. Виконують операцію XOR з першим байтом блоку даних і першим байтом M1. Вибирають ключ у таблиці ключів, розрахованої в раунді 1. Використовують обчислене значення XOR як індекс таблиці. Виконують операцію XOR з кожним, крім першого, байтом блоку даних і відповідним байтом обраного ключа.

b. Виконують операцію XOR із другим байтом блоку даних і другим байтом M1. Вибирають ключ у таблиці ключів, розрахованої в раунді 1. Використовують обчислене значення XOR як індекс таблиці. Виконаєте операцію XOR з кожним, крім другого, байтом блоку даних і відповідним байтом обраного ключа.

c. Продовжують ці дії з усім блоком даних (з 3-10 байтами), поки не буде використаний кожний байт для вибору ключа з таблиці після виконання операції XOR з ним і відповідним значенням M1. Потім виконують операцію XOR з кожним, крім використаного для вибору ключа, байтом, і ключем.

d. Повторюють етапи a-c для M2.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Інтерфейс головного вікна програми зображено на рисунку 5.1.

Для відображення поточного стану слугують наступні вікна, розташовані головним чином у лівій половині вікна:

– Відео з камери.

– Часова діаграма звуку.

– Відповідь з COM порту.

– Індикація заряду батареї.

Обробка голосових команд починається після натискання кнопки «Сказати».

Для запису нових команд слугує кнопка «Внести команду», а також «Передати» та «І додати до списку».

Для опису та редагування операцій, які виконуються після надходження голосових команд використовується блок «Керування голосовими командами», за допомогою якого встановлюються блок старту, блок зупинки та загальна кількість блоків.

Для кожної команди встановлюються параметри кута, кількості кроків та швидкості.

Список наявних команд відображається у правій стороні вікна. Там же можливе пряме керування робототехнічним пристроєм як за допомогою програмних кнопок, так і з використанням списку команд. При прямому керуванні сигнали керування та відповідь подаються через COM-порт.

Також в програма дозволяє зберігати записані команди для використання в наступних сесіях.

Кількість об’єктів керування, що відображається у головному вікні встановлюється за допомогою кнопки «Налаштування».

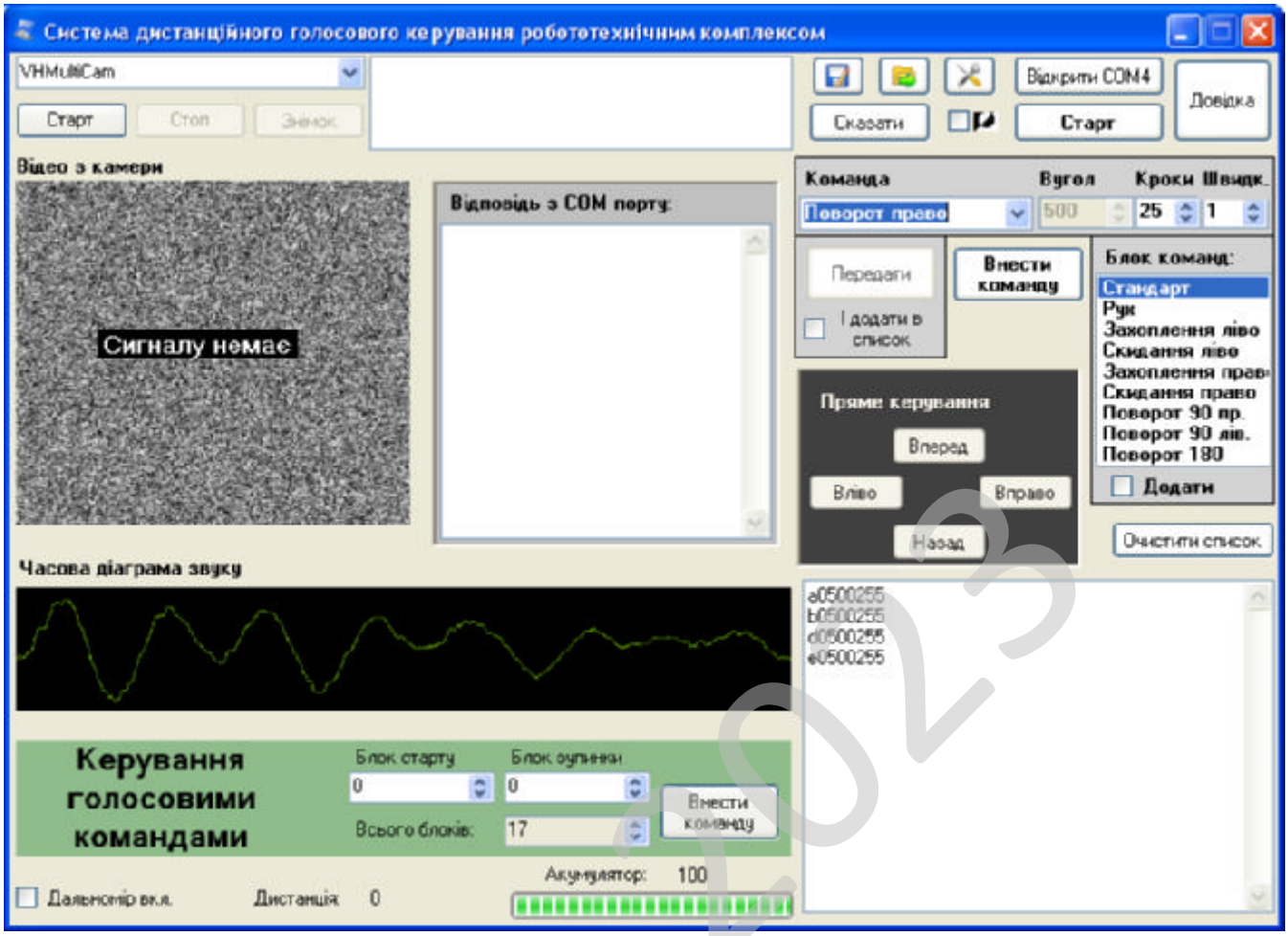

Рисунок 5.1 – Основне вікно програми

На рисунку 5.2 зображено вікно довідки про програму, в якому вказується тема магістерської роботи, її розробник, а також керівник.

Рисунок 5.2 – Вікно довідки

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи дистанційного голосового керування робототехнічним комплексом.

Метою розробки є дослідження та програмна реалізація системи дистанційного голосового керування робототехнічним комплексом.

Об'єктом дослідження є процес дистанційного голосового керування робототехнічним комплексом.

Предметом дослідження є методи дистанційного голосового керування робототехнічним комплексом.

Методи дослідження базуються на методах теорії Інтернету речей, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод дистанційного голосового керування робототехнічним комплексом.

– Розроблено вітчизняний продукт дистанційного голосового керування робототехнічним комплексом, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи дистанційного голосового керування робототехнічним комплексом. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	40
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження табл. 7.1

1	2	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	1
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	_	2
13. Рівень спрацьованості колективу (1-6)	_	2
14. Ступінь вимірності процесів (1-6)	_	3
15. Необхідна надійність програмного забезпечення (1-6)	_	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	_	2
17. Складність кінцевого програмного продукту (1-6)	_	2
18. Необхідний рівень забезпечення повторного використання (1-6)	_	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	_	2
20. Вимоги до швидкодії ПП (1-6)	_	2
21. Обмеження на розміри основного сховища даних (1-6)	_	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	_	2
23. Професійний рівень аналітиків (1-6)	_	2
24. Професійний рівень програмістів (1-6)	_	2
25. Постійність складу команди розробників (1-6)	_	2
26. Досвід розробки додатків (1-6)	_	2
27. Досвід роботи з обчислювальною платформою (1-6)	_	2

Продовження табл. 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	40000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	55
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \text{ Size}^B , \quad (7.1)$$

де A – коефіцієнт Боема, A=2,45; Size – загальний об'єм відлагодженого програмного коду, тис. рядків; B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \qquad (7.2)$$

де Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B=1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн}=T_{ном}\, ПВ_j, \qquad (7.3)$$

де $ПВj$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$
$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3CT_{уточн}^{0{,}33+0{,}2(B-1{,}01)}S, \qquad (7.4)$$

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3\cdot3{,}22\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot90=183 \text{ люд/день}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	183	Ф 7.1-7.4
Впровадження	13	Д13
Всього	224	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де Fpq – плановий фонд робочого часу одного спеціаліста, днів, Тпз – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{224 \cdot 1}{60\text{-}5} = 4 \text{ ставки}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	7	630	10,5
Монітор	60	7	420	7
Клавіатура	30	7	210	3,5
Маніпулятор «мишка»	30	7	210	3,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	3	90	1,5
Кабельні господарства ЛОМ на 1 м.п.	2,5	450	1125	18,75
Копіювальний апарат	140	2	280	4,67
Усього за рік:			$З_ч$	54,75

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2} \quad (7.6)$$

$$Ф_{др}^{с} = \frac{55 \cdot 3}{1,2} = 137,5 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}} \quad (7.7)$$

$Ч_{ел}$=137,5/(60·8)=0,3 ставки

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів – електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	Кількість штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, серверу доступу АДСЛ (ОС Linux), Wi-Fi налаштування ADSL, VPN, PPPoE, Frame Relay	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів д мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців:

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (ІТ-менеджер)	1	15000	45000
Продакт-менеджер	0,25	14000	10500
Інженер-програміст	4,0	14648,5	175782
Інженер-електронщик	0,3	14000	12600
Інженер-системотехнік	0,25	14000	10500
Адміністратор мережі	0,5	14000	21000
Системний програміст	0,25	14000	10500
Дизайнер WEB	0,25	14000	10500
Інженер-верстальник	0,25	14000	10500
Бухгалтер-економіст	0,5	14966	22449
Всього за період розробки	$R_{сп}$=7,55	-	$Ф_{роб}$=329331

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де Фроб – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{329331}{7,55 \cdot 60} = 727 \ грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі

$$Б_{уд} = R_{сп}^{1} S_{у} Ц_{пл}, \quad (7.9)$$

де $R_{сп}^{1}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць. Sy – питома площа на одне робоче місце, $м^2$; Цпл – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 37 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн на одне робоче місце. Тобто

$$I_{нв} = R_{сп}^{1} \cdot Ц_{м}, \quad (7.10)$$

де Цм – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 8 \cdot 3500 = 28000 \text{ грн}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались за прайсом фірми Компбест за 16.11.23 – джерело https://compbest.com.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core™ i3 10105 (BX8070110105) 1200, 4 ядра, 8 потоків, 3.7 GHz, 4.4 GHz, TDP – 65 Вт, 14nm, BOX	-
Системна плата	ASUS PRIME H510M-K сокет – 1200, DDR4, 3200 MHz, LAN – 1 Гбіт/с, D-Sub (VGA), HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	Intel UHD Graphics 630	-
Жорсткий диск	SSD M.2 2280 512GB LEVEN (JP600PCIE512GB) Серія – JP600, 512 GB, 3D TLC NAND, M.2, PCI Express 3.0 x4	-
Оперативна пам’ять	DDR4 8GB 3200 MHz Fury Beast Black Kingston Fury (ex.HyperX) (KF432C16BB/8)	-
Блок живлення	Gamemax 500W (GM-500B) ATX 12V v2.3, 500 Вт, 20+4 pin, CPU – 4+4pin, GPU – 1x6 pin, SATA – 3, Peripheral – 2, +12V1 – 20A, 1x120 мм, 150 x 140 x 86 мм	-
Корпус	Vinga CS210B, Miditower, ATX, Micro – ATX, Mini – ITX, Слотів расширення – 7	-
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-B, int. 3.5", 1*USB2.0+AUDIO+1394, multi: All Type Cards, black	220
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	LG W2363V-WF Wide LCD 2ms, 70 000:1, 300кд/м2, 170/160, D-Sub / Glossy White	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струменевий	Epson Stylus Photo P50 (C11CA45341) + USI cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5			
4. Вимірювальні пристрої	5190	-	-
5. Господарський інвентар	28000	-	-
Всього по групі	33190	25	8297,5
Нематеріальні активи			
6. Нематеріальні активи	40000	10	4000
Разом	$К_р$= 1596075		$А_р$= 140140

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де Ne – Кількість екземплярів програм, шт.

$$З_о = 727 \cdot 224/40 = 4071 \text{ грн}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 4071 \cdot 10 \cdot 0{,}01 = 407 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основної та додаткової зарплати

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де $Н_с$ – відрахування на соціальні потреби, %

$$С_{оц} = 0{,}01 \cdot 22(4071+407) = 985 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом Нг=15% від основної зарплати

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 4071 \cdot 15 \cdot 0{,}01 = 611 \text{ грн}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де $З_{М1}$ – вартість паперу, грн., $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн., $З_{М3}$ – вартість фарби, картриджів, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно виданих норм $n_{міс}$ приймаємо 1/6 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки $N_м$ =3 міс:

$$З_{М1} = Ц_п \cdot N_м \, n_{міс}. \quad (7.16)$$

$$З_{М1} = 210 \cdot 3 \cdot 1/6 = 105 \text{ грн}.$$

Згідно виданих норм до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків в кількості 10 примірників:

$$З_{М2} = \sum Ц_{д}., \quad (7.17)$$

де Цд – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 49,2 грн/шт., DVD-R LG 4,7Gb, 16x speed Cake box – 49,2 грн/шт.

$$З_{М2} = 49{,}2 \cdot 10 = 492 \text{ грн.}$$

Згідно виданих норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_{з}., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570=1702 \text{ грн.}$$

$$З_{М} = (105 + 492 + 1702)/40=57 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом (Нп = 15%) від основної зарплати виконавців

$$О_{п} = З_{о} \cdot Н_{п} \cdot 0{,}01, \quad (7.19)$$

де Нп – норматив витрат на освоєння нових мов програмування, %

$$О_{п} = 4071 \cdot 15 \cdot 0{,}01=611 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 40 прим.)

$$А_{м} = \frac{А_{р} \cdot N_{міс}}{N_{е} \cdot 12}, \quad (7.20)$$

де Ар – загальна річна сума амортизаційних відрахувань, грн.

$$А_{м} = 140140 \cdot 3/(40 \cdot 12)=876 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

$$С_{п} = З_{о} + З_{д} + С_{оц} + Г_{осп} + З_{м} + О_{п} + А_{м}. \quad (7.21)$$

$$C_n = 4071+407+985+611+57+611+876 = 7618 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (Рп) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 55%

$$П_р = 0{,}01 \cdot P_n \cdot C_n, \quad (7.22)$$

де Рс – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 55 \cdot 7618 = 4190 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1. Основна зарплата виконавців	$З_о$	4071
2. Додаткова зарплата виконавців	$З_д$	407
3. Відрахування на соціальні потреби	$С_{оц}$	985
4. Загальногосподарські витрати	$Г_{осп}$	611
5. Витрати на матеріали	$З_М$	57
6. Освоєння нових операційних систем, мов програмування	$О_п$	611
7. Амортизація основних фондів	$А_м$	876
8. Повна собівартість програмного забезпечення	$С_п$	7618
9. Плановий прибуток	$П_р$	4190
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	11808
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	2361,6
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	15998

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	15998
Всього капітальних витрат	–	15998

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на оплату праці	$З_р$	53690	33550
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	4000
Всього витрат за рік	$І$	53690	37550

Витрати на оплату праці:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \qquad (7.23)$$

де Тр – кількість годин обслуговування системи за рік, *год.*; Зг – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 400 годин на рік до 250 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\,баз} = 400 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 53680 \; грн.$$

до

$$З_{р\,нов} = 250 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 33550 \; грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності (Пел) в кіловатах, часу експлуатації технічних засобів (Тр) в годинах та ціни однієї кіловат-години (Цел).

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \qquad (7.24)$$

Після впровадження програмного продукту витрати на електроенергію не змінюються і складають Зел баз =·Зел нов

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	15998	–	3999,5
Всього відрахувань	-	–	15998	–	3999,5

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (11808 - 7618) \cdot 40 - (0,05 \cdot 1408000 + 0,5 \cdot 114885 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 40000) \cdot 3/12 = 132565 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р^*$ – балансова вартість основних фондів розробника.

$$T_в = \frac{1596075}{(11808 - 7618) \cdot 40 \cdot 12 / 3} = 2,4 \; років.$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н (K_н - K_б), \quad (7.27)$$

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (53690 - 37550) - 0,25 \cdot 15998 = 12141 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \quad (7.28)$$

$$T_{сп} = \frac{15998}{53690 \ - \ 37550} = 0{,}99 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	40
2. Повна собівартість розробленої програми	Грн	7618
3. Ціна розробленої програми	Грн.	11808
4. Плановий прибуток від реалізації розробленої програми	Грн.	4190
5. Рентабельність програмної продукції	%	55
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1596075
7. Загальний прибуток від реалізації програмної продукції	Грн.	167600
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	132565
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	2,4
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	15998
11.Величина економічного ефекту у користувача програмної продукції	Грн.	12141
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,99

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Сьогодення, а тим більше, майбутнє, вже важко уявити без комп'ютерів та іншої електронної техніки.

Законом України “Про охорону праці” [1] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207 [4], який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2].

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням [2]. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2], та «Правила

охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп’ютера на організм програміста визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

8.2 Шкідливі і небезпечні фактори при роботі з комп’ютером

Можна виділити наступні основні фактори, що впливають на стан здоров'я людей, які працюють за комп'ютером:

– сидяче положення на протязі тривалого періоду;

– вплив електромагнітного випромінювання монітора;

– втома очей, навантаження на зір;

– перевантаження суглобів кистей;

– стрес при втраті інформації.

У кожному з цих випадків ступінь ризику прямо пропорційний часу, що проводиться за комп’ютером і поблизу нього.В сучасних умовах взаємодія людини з технікою значно ускладнилась, що вимагає комплексного підходу, який передбачає розгляд людини, технічних засобів праці та виробничого середовища, як взаємозв'язаних елементів єдиної системи. Все вищесказане в повній мірі відноситься й до системи «людина–комп'ютер–середовище».

Вагомий вплив на працездатність та здоров’я користувачів комп’ютерів здійснює виробниче середовище [3]. Це середовище у виробничих приміщеннях (офісах), в основному, визначається мікрокліматом, освітленням, наявністю шкідливих речовин у повітрі, рівнем шуму, випромінювання.

Для того, щоб об’єктивно проаналізувати відповідність умов праці діючим нормативно-правовим актам та запропонувати заходи щодо зменшення негативного впливу комп’ютера на організм людини необхідно здійснити санітарно-гігієнічну характеристику умов працівника, який працює з програмним продуктом.

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	3
Довжина	6
Висота	3,4

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6
Обсяг, V	$м^3$	не менше 20.0	20,4

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин) [2].

У зазначеному приміщенні працює 3 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та а об’єм приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам

(Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови Головного державного санітарного лікаря України [5], робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 кКал. у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	21-23	40-58	0,11
Тепла	23-25	50-70	0,1	23-25	50-65	0,1

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер Canon LBP-6030B (8468B006), електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5 – 28 – 2006 р можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути

освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору).

8.5 Розрахункова частина

Завдання: розрахувати *штучне освітлення робочого приміщення.*

Початкові дані: ширина *робочого* приміщення: 3 м.; довжина – 6 м.; висота – 3,4 м.

Розрахунок штучного освітлення проведемо за методом коефіцієнта використання світлового потоку.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; *E = 300 Лк;*

S – площа освітлюваного приміщення (у нашому випадку *S=3 x 6 = 18* м^2);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку *Z = 1,1*);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку *K = 1,5);*

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50% *[6]*.

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де:

S – площа приміщення, *S = 18* м^2;

h – розрахункова висота підвісу, *h = 3,4* м;

A – ширина приміщення, А = 3 м;

B – довжина приміщення, В = *6* м.

Підставимо всі значення у формулу та визначемо індекса приміщення: *i=0,57.*

Знаючи індекс приміщення, за знаходимо *n = 0.29 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників відповідного типу).* Підставимо всі значення у формулу, визначемо світловий потік: *F=19370 Лм.*

Для штучного освітлення приміщення використовуються *LED панель MAXUS ASSISTANCE PRO 40W 5000K WHITE (M1051440531)*, світловий потік яких $F_л = 4000$ Лм.

Число світильників визначається по формулі:

$$N=F/F_л$$

де: *F* – світловий потік,

$F_л$ – світловий потік одного *світильника*.

$$N= 19370/ 4000=4,84 \text{ шт.}$$

Приймаємо необхідну кількість світильників 5 шт.

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок захисного штучного освітлення. Розроблено заходи з охорони праці.

Список використаних джерел інформації

1. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

2. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2.007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

3. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

4. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

5. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

6. Оришака О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький: ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення: 16.06.2023).

7. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальностей 123 "Комп'ютерна інженерія" та 122 "Комп'ютерні науки" / Оришака О. В., Марченко К.М.; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення. – Кропивницький : ЦНТУ, 2022. — 19 с. [Електронний ресурс]. – Режим доступу (URI): http://dspace.kntu.kr.ua/jspui/handle/123456789/12240 (дата звернення: 16.06.2023).

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи дистанційного голосового керування робототехнічним комплексом.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів дистанційного голосового керування робототехнічним комплексом.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем дистанційного голосового керування робототехнічним комплексом.

– Досліджена система дистанційного голосового керування робототехнічним комплексом.

– На основі отриманих результатів досліджень створена програмна реалізація системи дистанційного голосового керування робототехнічним комплексом.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання дистанційного голосового керування робототехнічним комплексом.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Visual C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм REDOC III.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 12141 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,99 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гуровський В.Д. Дослідження та програмна реалізація системи дистанційного голосового керування робототехнічним комплексом // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. Peter Hoddie, Lizzie Prader «IoT Development for ESP32 and ESP8266 with JavaScript: A Practical Guide to XS and the Moddable SDK» ISBN-13 (pbk): 978-1-4842-5069-3 ISBN-13 (electronic): 978-1-4842-5070-9
3. STM32CubeMX for STM32 configuration and initialization C code generation. User manual. June 2022. 397 p.
4. І.В.Чихіра, А.Г. Микитишин Конспект лекцій з дисципліни «Програмування систем реального часу» / Укладачі : Чихіра І.В., Микитишин А.Г., – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя , 2016. – 76 с.
5. Jack Ganssle and Michael Barr. 2003. Embedded Systems Dictionary. CMP Books.
6. Технології інтернету речей. Навчальний посібник [Електронний ресурс]: / Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 271 с.
7. Greg Dunko, Joydeep Misra, Josh Robertson, Tom Snyder “A reference guide to the Internet of Things” / 2017 Bridgera LLC, RIoT.
8. Donald Norris “Programming with STM32. Getting started with the Nucleo Board and C/C++” 416 p. 2018.
9. Neil Kolban “Kolban’s book on ESP32”. Texas, USA. 951 p.
10. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises».

Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

11. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

12. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

13. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

14. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

15. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

16. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T.**, «**Representation of Cascade Codes in the Frequency Domain**».** In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. **Springer**, Cham. 2021. pp 557-587.

17. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

18. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

19. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

20. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

21. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

22. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

23. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

24. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

25. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

26. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

27. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

28. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

29. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

30. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

31. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-

телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

32. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

33. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

34. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

35. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

36. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

37. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

38. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

39. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

40. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

41. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

42. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

43. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

44. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

45. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

46. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

47. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

48. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп'ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

49. Смірнов О.А., Стасєв Ю.В. Бараннік В.В. Захист інформації в автоматизованих системах управління. Навчальний посібник – Харків: ХУПС, 2015. – 264 с.

50. Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. О.А. Смірнова Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 17.04.2012 року № 1/11-5249. – Кіровоград: КНТУ 2012. – 250 с.

51. Смірнов О.А., Євсеєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп'ютерних мереж. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія» та 8.0925 «Автоматизація й комп'ютерно-інтегровані технології». За ред. О.А. Смірнова Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки, молоді та спорту України від 1.12.2011 року № 1/11-11258. – Кіровоград: КНТУ 2012. – 454 с

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи дистанційного голосового керування робототехнічним комплексом.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 34-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи дистанційного голосового керування робототехнічним комплексом.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи дистанційного голосового керування робототехнічним комплексом;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Visual C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий розрахунок штучного освітлення робочого приміщення.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 108 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 12.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Коваленко О.В.

Дослідження та програмна реалізація системи дистанційного голосового керування робототехнічним комплексом

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 47

Літера: РП

Кропивницький – 2023 року

// main.cpp – основна програма

```
#include "WordRecognizer.h"

#include <exception>
#include <iostream>
#include <fstream>
#include <string>
#include <vector>

#include <float.h>
#include <math.h>

using namespace std;
using namespace WordRecognizer;

//! Реалізація інтерфейсу IVectorsSet із зберіганням всіх даних в оперативній
пам'яті
class MemoryVectorsSet : public IVectorsSet
{
private:
	vector<vector<float> > m_data;
	int m_dim;

public:
	MemoryVectorsSet() : m_dim(0)
	{
	}

	void SetDim(const int &_dim)
	{
		if (_dim < 0)
			throw exception("MemoryVectorsSet::SetDim: _dim < 0");
		m_dim = _dim;
	}

	const int GetDim() const
	{
		return m_dim;
	}

	void SetSize(const int &_size)
	{
		if (_size < 0)
			throw exception("MemoryVectorsSet::SetSize: _size < 0");
		m_data.resize(_size);
	}

	const int GetSize() const
	{
		return (const int)m_data.size();
	}

	void SetVectorsNum(const int &_indx, const int &_vectorsNum)
	{
		if (_indx >= GetSize())
			throw exception("MemoryVectorsSet::SetVectorsNum: _indx >=
size");
		if (m_dim <= 0)
			throw exception("MemoryVectorsSet::SetVectorsNum: m_dim <=
0");
		m_data[_indx].resize(_vectorsNum * m_dim);
		memset(&m_data[_indx][0], 0, sizeof(float) * _vectorsNum * m_dim);
	}

	void GetVectors(
		const int		&_indx,
```

```
            int                     &_vectorsNum,
            const float*      &_vectors) const
      {
            if (_indx >= GetSize())
                  throw exception("MemoryVectorsSet::GetVectors: _indx >=
size");
            if (m_dim <= 0)
                  throw exception("MemoryVectorsSet::GetVectors: m_dim <= 0");
            _vectorsNum = (int)m_data[_indx].size() / m_dim;
            _vectors = (_vectorsNum > 0) ? &m_data[_indx][0] : NULL;
      }

      void GetVectors(
            const int   &_indx,
            int               &_vectorsNum,
            float*            &_vectors)
      {
            const float* pVectors;
            GetVectors(_indx, _vectorsNum, pVectors);
            _vectors = const_cast<float*>(pVectors);
      }


};

//! Клас-оболонка для спрощення роботи з системою розпізнавання слів
class WordRecognizerWrapper
{
public:
      vector<vector<string> > m_files;
      vector<string> m_wordsBase;
      MemoryVectorsSet m_mfcc;

      int m_dim;
      int meansNum;
      vector<float> m_means;

      float m_adaptWeight;
      MemoryVectorsSet m_adaptMeans;

      int m_LDASize;
      vector<float> m_LDAMatrix;
      vector<float> m_mean;

      MemoryVectorsSet m_projections;

      vector<string> m_learnWords;
      vector<float> m_wordsMean;
      vector<float> m_wordsSigma;

      bool m_baseOK;
      bool m_meansOK;
      bool m_adaptMeansOK;
      bool m_LDAOK;
      bool m_projectionsOK;
      bool m_compareSystemOK;

public:
      WordRecognizerWrapper() : m_baseOK(false), m_meansOK(false),
m_adaptMeansOK(false), m_LDAOK(false),
            m_projectionsOK(false), m_compareSystemOK(false), meansNum(0),
m_adaptWeight(50.), m_LDASize(0)
      {
      }

      ~WordRecognizerWrapper()
      {
      }
```

```
	void LoadBase(const char* _baseFile)
	{
		string fileName(_baseFile);
		string path = fileName.substr(0, fileName.find_last_of("/\\"));
		path += "/";
		ifstream baseInfo(_baseFile);
		if (!baseInfo.is_open())
			throw exception("неможливо відкрити базовий файл");

		const int maxStr = 5000;
		char tmpStr[maxStr];
		int wordsNum = 0;

		baseInfo.getline(tmpStr, maxStr);
		wordsNum = atoi(tmpStr);
		if (wordsNum <= 0)
			throw exception("кількість слів <= 0");

		m_wordsBase.resize(wordsNum);
		m_files.resize(wordsNum);

		int filesNum = 0;
		int filesCount = 0;

		for (int i = 0; i < wordsNum; i++)
		{
			baseInfo.getline(tmpStr, maxStr);
			m_wordsBase[i] = tmpStr;
			if (m_wordsBase[i].length() == 0)
				ThrowFormatException("довжина слова [%d] == 0", i);
			baseInfo.getline(tmpStr, maxStr);
			filesNum = atoi(tmpStr);
			if (filesNum <= 0)
				ThrowFormatException("кількість файлів для слова %s <=
0", m_wordsBase[i].c_str());
			m_files[i].resize(filesNum);
			filesCount += filesNum;
			for (int j = 0; j < filesNum; j++)
			{
				if (baseInfo.eof())
					ThrowFormatException("неочікуваний кінець файла
всередині слова %s", m_wordsBase[i].c_str());
				baseInfo.getline(tmpStr, maxStr);
				m_files[i][j] = tmpStr;
				if (m_files[i][j].length() == 0)
					ThrowFormatException("довжина імені файлу [%d] для
слова %s == 0", j, m_wordsBase[i].c_str());
			}
		}
		baseInfo.close();

		m_mfcc.SetSize(filesCount);
		filesCount = 0;
		for (int i = 0; i < wordsNum; i++)
		{
			for (size_t j = 0; j < m_files[i].size(); j++)
			{
				ifstream fileMFCC((path + m_files[i][j]).c_str(),
ios_base::in | ios_base::binary);
				if (!fileMFCC.is_open())
					ThrowFormatException("неможливо відкрити файл %s",
(path + m_files[i][j]).c_str());

				// Читання файлу SPRO
				unsigned short dim = 0;
				long flag = 0;
				float freq = 0;

				fileMFCC.seekg(0, ios_base::end);
```

```
                        size_t fileSize = (size_t)fileMFCC.tellg();
                        fileMFCC.seekg(0, ios_base::beg);
                        if (fileSize < 10)
                              ThrowFormatException("неможливо прочитати файл
%s", (path + m_files[i][j]).c_str());

                        fileMFCC.read((char*)&dim, sizeof(dim));
                        fileMFCC.read((char*)&flag, sizeof(flag));
                        fileMFCC.read((char*)&freq, sizeof(freq));

                        if ((fileSize - 10) % (sizeof(float) * dim) != 0)
                              ThrowFormatException("неможливо прочитати файл
%s", (path + m_files[i][j]).c_str());

                        if (m_mfcc.GetDim() == 0)
                              m_mfcc.SetDim((int)dim);
                        else if (m_mfcc.GetDim() != (int)dim)
                              ThrowFormatException("файл %s має неправильну
розмірність: %d", (path + m_files[i][j]).c_str(), (int)dim);

                        int vectorsNum = (int)((fileSize - 10) / (sizeof(float)
* dim));
                        if (vectorsNum == 0)
                              ThrowFormatException("файл %s пустий");

                        m_mfcc.SetVectorsNum(filesCount, vectorsNum);
                        float* pVectors;
                        m_mfcc.GetVectors(filesCount, vectorsNum, pVectors);
                        fileMFCC.read((char*)pVectors, sizeof(float) *
vectorsNum * m_mfcc.GetDim());
                        filesCount++;

                        if (fileMFCC.fail())
                              ThrowFormatException("неможливо прочитати файл
%s", (path + m_files[i][j]).c_str());
                        fileMFCC.close();
                  }
            }
            m_baseOK = true;
      }

      ResultStatus CreateMeans(const int &_maxItersNum, const int &_meansNum,
const int &_minVectorsNum)
      {
            if (!m_baseOK) throw exception("спершу завантажте базу");
            if (_meansNum <= 0) throw exception("кількість середніх <= 0");

            meansNum = _meansNum;
            m_dim = m_mfcc.GetDim();
            m_means.resize(meansNum * m_dim);
            ResultStatus result = WordRecognizer::KMeans(m_mfcc, _maxItersNum,
meansNum, _minVectorsNum, &m_means[0]);
            m_meansOK = true;
            return result;
      }

      void SetAdaptWeight(const float &_weight)
      {
            m_adaptWeight = _weight;
      }

      void MakeAdaptation()
      {
            if (!m_baseOK) throw exception("спершу завантажте базу");
            if (!m_meansOK || (m_means.size() == 0)) throw exception("середні не
створено");

            int superVectorDim = meansNum * m_mfcc.GetDim();
```

```
            if ((int)m_means.size() != superVectorDim) throw
exception("неправильний розмір середніх");

            // Групуємо адаптовані середні значення по приналежності одному
слову
            m_adaptMeans.SetDim(superVectorDim);
            m_adaptMeans.SetSize((int)m_files.size());
            int count = 0;
            for (int i = 0; i < m_adaptMeans.GetSize(); i++)
            {
                  m_adaptMeans.SetVectorsNum(i, (int)m_files[i].size());
                  count += (int)m_files[i].size();
            }
            if (count != m_mfcc.GetSize()) throw exception("m_mfcc.GetSize() !=
sum of m_files[i].size()");

            // Проводимо адаптацію для всіх файлів
            count = 0;
            int adaptVectorsNum = 0;
            float* adaptResult = NULL;
            int mfccVectorsNum = 0;
            float* mfccData = NULL;
            for (int i = 0; i < m_adaptMeans.GetSize(); i++)
            {
                  m_adaptMeans.GetVectors(i, adaptVectorsNum, adaptResult);
                  for (int j = 0; j < adaptVectorsNum; j++, adaptResult +=
m_adaptMeans.GetDim(), count++)
                  {
                        m_mfcc.GetVectors(count, mfccVectorsNum, mfccData);
                        WordRecognizer::Adaptation(m_mfcc.GetDim(),
mfccVectorsNum, meansNum, &m_means[0], m_adaptWeight, mfccData, adaptResult);
                  }
            }
            m_adaptMeansOK = true;
      }

      ResultStatus CreateLDA(const int &_LDASize)
      {
            if (!m_adaptMeansOK) throw exception("спершу виконайте адаптацію");

            m_LDASize = _LDASize;
            int dim = m_adaptMeans.GetDim();
            if (dim <= 0) throw exception("розмір адаптованих середніх <= 0");
            m_LDAMatrix.resize(m_LDASize * dim);
            m_mean.resize(dim);
            ResultStatus result = WordRecognizer::CreateLDAMatrix(m_adaptMeans,
m_LDASize, &m_LDAMatrix[0], &m_mean[0]);

            m_LDAOK = true;
            return result;
      }

      void MakeProjections()
      {
            if (!m_adaptMeansOK) throw exception("спершу виконайте адаптацію");
            if (!m_LDAOK) throw exception("створіть спершу LDA");

            m_projections.SetDim(m_LDASize);
            m_projections.SetSize(m_adaptMeans.GetSize());
            int adaptDim = m_adaptMeans.GetDim();

            for (int i = 0; i < m_adaptMeans.GetSize(); i++)
            {
                  int vectorsNum = 0;
                  float* pAdaptVectors = NULL;
                  float* pProjectionVectors = NULL;
                  m_adaptMeans.GetVectors(i, vectorsNum, pAdaptVectors);
                  m_projections.SetVectorsNum(i, vectorsNum);
                  m_projections.GetVectors(i, vectorsNum, pProjectionVectors);
```

```
                  for (int j = 0; j < vectorsNum; j++, pAdaptVectors +=
adaptDim, pProjectionVectors += m_LDASize)
                        WordRecognizer::MakeProjection(m_LDASize,
m_adaptMeans.GetDim(), &m_LDAMatrix[0], &m_mean[0],  pAdaptVectors,
pProjectionVectors);
            }
            m_projectionsOK = true;
      }

      ResultStatus CreateCompareSystem(const float &_minSigma)
      {
            if (!m_baseOK) throw exception("спершу завантажте базу");
            if (!m_projectionsOK) throw exception("виконайте спершу проекції");
            int dim = m_projections.GetDim();
            int wordsNum = m_projections.GetSize();

            if (wordsNum != (int)m_wordsBase.size())
                  throw exception("число слів у базі != числу блоків проекцій");

            m_learnWords = m_wordsBase;
            m_wordsMean.resize(wordsNum * dim);
            m_wordsSigma.resize(wordsNum);
            ResultStatus result =
WordRecognizer::CreateCompareSystem(m_projections, &m_wordsMean[0],
&m_wordsSigma[0], _minSigma);
            m_compareSystemOK = true;
            return result;
      }

      void SaveSystem(const char* _fileName)
      {
            ofstream stateFile(_fileName, ios_base::out | ios_base::binary |
ios_base::trunc);
            if (!stateFile.is_open())
                  ThrowFormatException("неможливо відкрити файл %s", _fileName);

            // середні
            char boolChar = (m_meansOK) ? 0x01 : 0x00;
            stateFile.write(&boolChar, sizeof(boolChar));
            stateFile.write((char*)&m_dim, sizeof(m_dim));
            stateFile.write((char*)&meansNum, sizeof(meansNum));
            if (meansNum * m_dim > 0)
                  stateFile.write((char*)&m_means[0], sizeof(m_means[0]) *
meansNum * m_dim);

            // адаптація
            stateFile.write((char*)&m_adaptWeight, sizeof(m_adaptWeight));

            // LDA
            boolChar = (m_LDAOK) ? 0x01 : 0x00;
            stateFile.write(&boolChar, sizeof(boolChar));
            stateFile.write((char*)&m_LDASize, sizeof(m_LDASize));
            if (m_LDASize * m_dim * meansNum > 0)
            {
                  stateFile.write((char*)&m_LDAMatrix[0], sizeof(m_LDAMatrix[0])
* m_LDASize * m_dim * meansNum);
                  stateFile.write((char*)&m_mean[0], sizeof(m_mean[0]) * m_dim *
meansNum);
            }

            // порівняння систем
            boolChar = (m_compareSystemOK) ? 0x01 : 0x00;
            stateFile.write(&boolChar, sizeof(boolChar));
            int wordsNum = m_learnWords.size();
            stateFile.write((char*)&wordsNum, sizeof(wordsNum));
            if (wordsNum > 0)
            {
                  for (int i = 0; i < wordsNum; i++)
```

```
                {
                        int len = m_learnWords[i].length();
                        stateFile.write((char*)&len, sizeof(len));
                        stateFile.write(m_learnWords[i].c_str(), len);
                }
                stateFile.write((char*)&m_wordsMean[0], sizeof(m_wordsMean[0])
* wordsNum * m_LDASize);
                stateFile.write((char*)&m_wordsSigma[0],
sizeof(m_wordsSigma[0]) * wordsNum);
        }

        if (stateFile.fail())
                ThrowFormatException("помилка запису файлу %s", _fileName);
    }

    void LoadSystem(const char* _fileName)
    {
        ifstream stateFile(_fileName, ios_base::in | ios_base::binary);
        if (!stateFile.is_open())
                ThrowFormatException("неможливо відкрити файл %s", _fileName);

        // середні
        char boolChar;
        stateFile.read(&boolChar, sizeof(boolChar));
        m_meansOK = (boolChar != 0);
        stateFile.read((char*)&m_dim, sizeof(m_dim));
        stateFile.read((char*)&meansNum, sizeof(meansNum));
        m_means.resize(meansNum * m_dim);
        if (meansNum * m_dim > 0)
                stateFile.read((char*)&m_means[0], sizeof(m_means[0]) *
meansNum * m_dim);

        // адаптація
        stateFile.read((char*)&m_adaptWeight, sizeof(m_adaptWeight));

        // LDA
        stateFile.read(&boolChar, sizeof(boolChar));
        m_LDAOK = (boolChar != 0);
        stateFile.read((char*)&m_LDASize, sizeof(m_LDASize));
        m_LDAMatrix.resize(m_LDASize * m_dim * meansNum);
        m_mean.resize(m_dim * meansNum);
        if (m_LDASize * m_dim * meansNum > 0)
        {
                stateFile.read((char*)&m_LDAMatrix[0], sizeof(m_LDAMatrix[0])
* m_LDASize * m_dim * meansNum);
                stateFile.read((char*)&m_mean[0], sizeof(m_mean[0]) * m_dim *
meansNum);
        }

        // порівняння систем
        stateFile.read(&boolChar, sizeof(boolChar));
        m_compareSystemOK = (boolChar != 0);
        int wordsNum = 0;
        stateFile.read((char*)&wordsNum, sizeof(wordsNum));
        m_learnWords.resize(wordsNum);
        m_wordsMean.resize(wordsNum * m_LDASize);
        m_wordsSigma.resize(wordsNum);
        if (wordsNum > 0)
        {
                vector<char> tmpStr;
                for (int i = 0; i < wordsNum; i++)
                {
                        int len = 0;
                        stateFile.read((char*)&len, sizeof(len));
                        tmpStr.resize(len + 1);
                        stateFile.read(&tmpStr[0], len);
                        tmpStr[len] = 0x00;
                        m_learnWords[i] = &tmpStr[0];
                }
```

```
                stateFile.read((char*)&m_wordsMean[0], sizeof(m_wordsMean[0])
* wordsNum * m_LDASize);
                stateFile.read((char*)&m_wordsSigma[0],
sizeof(m_wordsSigma[0]) * wordsNum);
          }

          if (stateFile.fail())
                ThrowFormatException("помилка читання файлу %s", _fileName);
     }

     void Test(const char* _resultsFileName, bool _useSigma, double &_wer)
     {
          if (!m_baseOK) throw exception("спершу завантажте базу");
          if (!m_projectionsOK) throw exception("виконайте спершу проекції");
          if (!m_compareSystemOK) throw exception("виконайте спершу порівняння
систем");

          ofstream resultsFile(_resultsFileName);
          if (!resultsFile.is_open())
                ThrowFormatException("неможливо відкрити файл %s",
_resultsFileName);

          int learnedWordsNum = (int)m_learnWords.size();
          resultsFile << "слова: ";
          for (int i = 0; i < learnedWordsNum; i++)
                resultsFile << "\t" << m_learnWords[i];
          resultsFile<<endl;

          int correctNum = 0;
          int attemptsNum = 0;
          vector<int> answers(learnedWordsNum);
          vector<float> probability(learnedWordsNum);
          int dim = m_projections.GetDim();
          for (int i = 0; i < m_projections.GetSize(); i++)
          {
                string testWordName = m_wordsBase[i];
                float* pTestVector = NULL;
                int vectorsNum = 0;
                m_projections.GetVectors(i, vectorsNum, pTestVector);
                for (int k = 0; k < learnedWordsNum; k++)
                      answers[k] = 0;
                for (int j = 0; j < vectorsNum; j++, pTestVector += dim)
                {
                      attemptsNum++;
                      WordRecognizer::GetProbability(dim, learnedWordsNum,
&m_wordsMean[0], &m_wordsSigma[0], pTestVector, _useSigma, &probability[0]);
                      float maxP = 0.;
                      int maxPindx = 0;
                      for (int k = 0; k < learnedWordsNum; k++)
                      {
                            if (probability[k] > maxP)
                            {
                                  maxP = probability[k];
                                  maxPindx = k;
                            }
                      }
                      answers[maxPindx]++;
                      if (m_learnWords[maxPindx].compare(testWordName) == 0)
                            correctNum++;
                }
                resultsFile << testWordName << ": ";
                for (int k = 0; k < learnedWordsNum; k++)
                      resultsFile << "\t" << answers[k];
                resultsFile << endl;
          }

          resultsFile << endl;
          resultsFile << "спроби num: " << attemptsNum << endl;
          resultsFile << "коректно num: " << correctNum << endl;
```

```
            _wer = 100. * (double)(attemptsNum - correctNum) / attemptsNum;
            resultsFile << "кількість помилок у слові: " << _wer << " %" <<
endl;
      }

};

struct Params
{
      string baseFile;              //!< Файл з описом бази (тестування або
навчання)
      string systemFile;                  //!< Файл, який зберігає результат
навчання системи
      string testResultsFile;       //!< Файл з результатами тестування
      bool learn;                         //!< Навчати систему?
      int meansNum;                       //!< Кількість середніх значень
      int kmeansMaxIters;                 //!< Максимальне число ітерацій
алгоритму K-середніх
      int minVectorsNumForKMeans;   //!< Найменше число векторів, яке відповідає
одному середньому значенню
      float adaptWeight;                  //!< "Чутливість" адаптації
      int LDASize;                        //!< Розмірність LDA-матриці
      float sigmaFloor;             //!< Мінімальне значення СКО
      bool useSigma;                      //!< Використовувати нормування на
внутрікласове СКО при тестуванні
      Params()
      {
            learn = false;
            meansNum = 10;
            kmeansMaxIters = 2000;
            minVectorsNumForKMeans = 10;
            adaptWeight = 20.f;
            LDASize = 20;
            sigmaFloor = 0.001f;
            useSigma = false;
      }
};

void PrintHelp(Params &_params)
{
      cout << "–help           Показати повний список параметрів." << endl <<
endl;
      cout << "--base          Файл з описом бази (обов'язковий). Формат файлу:"
<< endl;
      cout << "                [КІЛЬКІСТЬ СЛІВ]" << endl;
      cout << "                [WORD 1 NAME]" << endl;
      cout << "                [FILES NUM OF WORD 1]" << endl;
      cout << "                [FILE 1 OF WORD 1]" << endl;
      cout << "                [FILE 2 OF WORD 1]" << endl;
      cout << "                ............" << endl;
      cout << "                [WORD 2 NAME]" << endl;
      cout << "                [FILES NUM OF WORD 2]" << endl;
      cout << "                [FILE 1 OF WORD 2]" << endl;
      cout << "                [FILE 2 OF WORD 2]" << endl;
      cout << "                ............" << endl << endl;
      cout << "--system        Файл який містить результати навчання
(обов'язковий)." << endl << endl;
      cout << "--test_results  Файл з результатами тестування (обов'язковий)."
<< endl << endl;
      cout << "--learn         Навчання ("; if (_params.learn) cout <<
"enabled"; else cout << "disabled"; cout <<" by default)." << endl << endl;
      cout << "--means         Кількість середніх (" << _params.meansNum << "
by default)." << endl << endl;
      cout << "--max_iters     Максимальна кількість ітерацій для алгоритму K-
середніх (" << _params.kmeansMaxIters << " by default)." << endl << endl;
      cout << "--min_vectors   Мінімальна кількість векторів, які відповідають
одному середньому (" << _params.minVectorsNumForKMeans << " by default)." <<
endl << endl;
```

```
	cout << "--adapt_weight   Вага адаптації (" << _params.adaptWeight << " by
default)." << endl << endl;
	cout << "--lda           розмір LDA-матриці (" << _params.LDASize << " by
default)." << endl << endl;
	cout << "--use_sigma      Використання внутрішньо-класової нормалізації
відхилень при тестуванні ("; if (_params.useSigma) cout << "enabled"; else cout
<< "disabled"; cout << " by default)." << endl << endl;
	cout << "--sigma_floor    Стандартне мінімальне значення відхилень (" <<
_params.sigmaFloor << " by default)." << endl << endl;

}

bool ReadCommandLine(int argc, char* argv[],
	Params &_params)
{
	for (int arg = 1; arg < argc; arg++)
	{
		if (strcmp(argv[arg], "--help") == 0)
		{
			PrintHelp(_params);
			return false;
		}
		else if (strcmp(argv[arg], "--base") == 0)
		{
			if (argc <= arg + 1)
			{
				cout << "Не вдалося прочитати значення для параметра '--
base'" << endl;
				return false;
			}
			arg++;
			_params.baseFile = argv[arg];
		}
		else if (strcmp(argv[arg], "--system") == 0)
		{
			if (argc <= arg + 1)
			{
				cout << "Не вдалося прочитати значення для параметра '--
system'" << endl;
				return false;
			}
			arg++;
			_params.systemFile = argv[arg];
		}
		else if (strcmp(argv[arg], "--test_results") == 0)
		{
			if (argc <= arg + 1)
			{
				cout << "Не вдалося прочитати значення для параметра '--
test_results'" << endl;
				return false;
			}
			arg++;
			_params.testResultsFile = argv[arg];
		}
		else if (strcmp(argv[arg], "--learn") == 0)
		{
			_params.learn = true;
		}
		else if (strcmp(argv[arg], "--means") == 0)
		{
			if (argc <= arg + 1)
			{
				cout << "Не вдалося прочитати значення для параметра '--
means'" << endl;
				return false;
			}
			arg++;
			_params.meansNum = atoi(argv[arg]);
```

```
            }
            else if (strcmp(argv[arg], "--max_iters") == 0)
            {
                  if (argc <= arg + 1)
                  {
                        cout << "Не вдалося прочитати значення для параметра '--
max_iters'" << endl;
                        return false;
                  }
                  arg++;
                  _params.kmeansMaxIters = atoi(argv[arg]);
            }
            else if (strcmp(argv[arg], "--min_vectors") == 0)
            {
                  if (argc <= arg + 1)
                  {
                        cout << "Не вдалося прочитати значення для параметра '--
min_vectors'" << endl;
                        return false;
                  }
                  arg++;
                  _params.minVectorsNumForKMeans = atoi(argv[arg]);
            }
            else if (strcmp(argv[arg], "--adapt_weight") == 0)
            {
                  if (argc <= arg + 1)
                  {
                        cout << "Не вдалося прочитати значення для параметра '--
adapt_weight'" << endl;
                        return false;
                  }
                  arg++;
                  _params.adaptWeight = (float)atof(argv[arg]);
            }
            else if (strcmp(argv[arg], "--lda") == 0)
            {
                  if (argc <= arg + 1)
                  {
                        cout << "Не вдалося прочитати значення для параметра '--
lda'" << endl;
                        return false;
                  }
                  arg++;
                  _params.LDASize = atoi(argv[arg]);
            }
            else if (strcmp(argv[arg], "--use_sigma") == 0)
            {
                  _params.useSigma = true;
            }
            else if (strcmp(argv[arg], "--sigma_floor") == 0)
            {
                  if (argc <= arg + 1)
                  {
                        cout << "Не вдалося прочитати значення для параметра '--
floor'" << endl;
                        return false;
                  }
                  arg++;
                  _params.sigmaFloor = (float)atof(argv[arg]);
            }
            else
            {
                  cout << "Can't parse parameter " << argv[arg] << endl;
                  return false;
            }
      }
      if (_params.baseFile.length() == 0)
      {
```

```
            cout << "Parameter '--base’ не знайдено. Використайте ‘--help’ для
повного списку параметрів." << endl;
            return false;
      }
      if (_params.systemFile.length() == 0)
      {
            cout << "Parameter '--system' не знайдено. використайте '--help' для
повного списку параметрів." << endl;
            return false;
      }
      if (_params.baseFile.length() == 0)
      {
            cout << "Parameter '--base’ не знайдено. Використайте ‘--help’ для
повного списку параметрів." << endl;
            return false;
      }
      return true;
}

int main(int argc, char* argv[])
{

      Params params;
      if (!ReadCommandLine(argc, argv, params))
            return -1;

      try
      {
            ResultStatus status(RS_SUCCESS);
            WordRecognizerWrapper wrapper;
            double WER = 100;

            if (params.learn)
            {
                  cout << "Load learn base... ";
                  wrapper.LoadBase(params.baseFile.c_str());
                  cout << "OK" << endl;

                  cout << "Create means... ";
                  status = wrapper.CreateMeans(params.kmeansMaxIters,
params.meansNum, params.minVectorsNumForKMeans);
                  if (status != RS_SUCCESS)
                  {
                        cout << endl << «Увага: " <<
GetMessageFromStatus(status) << endl;
                  }
                  cout << "OK" << endl;


                  cout << "Адаптація... ";
                  wrapper.SetAdaptWeight(params.adaptWeight);
                  wrapper.MakeAdaptation();
                  cout << "OK" << endl;

                  cout << "LDA... ";
                  status = wrapper.CreateLDA(params.LDASize);
                  if (status != RS_SUCCESS)
                  {
                        cout << endl << «Увага: " <<
GetMessageFromStatus(status) << endl;
                  }
                  cout << "OK" << endl;

                  cout << "Виконати проекції... ";
                  wrapper.MakeProjections();
                  cout << "OK" << endl;

                  cout << "Виконати порівняння системи... ";
                  status = wrapper.CreateCompareSystem(params.sigmaFloor);
```

```
                cout << "OK" << endl;

                cout << "Зберегти... ";
                wrapper.SaveSystem(params.systemFile.c_str());
                cout << "OK" << endl;

                cout << "Тестувати базу навчання... ";
                wrapper.Test(params.testResultsFile.c_str(), params.useSigma,
WER);
                cout << "OK" << endl;
            }
            else
            {
                cout << "Завантажити... ";
                wrapper.LoadSystem(params.systemFile.c_str());
                cout << "OK" << endl;

                cout << "Завантажити тестову базу... ";
                wrapper.LoadBase(params.baseFile.c_str());
                cout << "OK" << endl;

                cout << "Адаптація... ";
                wrapper.MakeAdaptation();
                cout << "OK" << endl;

                cout << "Виконати проекції... ";
                wrapper.MakeProjections();
                cout << "OK" << endl;

                cout << "Тестувати базу навчання... ";
                wrapper.Test(params.testResultsFile.c_str(), params.useSigma,
WER);
                cout << "OK" << endl;
            }
            cout << "Помилка верифікації: " << WER << " %" << endl;
      }
      catch(std::bad_alloc &_e)
      {
            cout << endl << "Знайдено std::bad_alloc: " << _e.what() << endl;
            return -1;
      }
      catch(std::exception &_e)
      {
            cout << endl << "Знайдено std::exception: " << _e.what() << endl;
            return -1;
      }
      return 0;
}
```

// WordRecognizer.cpp – модуль розпізнавання голосових команд

```
#include "WordRecognizer.h"

#include <exception>
#include <vector>
#include <float.h>
#include <math.h>
#include <stdarg.h>

#include "alglib/dataanalysis.h"

using namespace std;
using namespace WordRecognizer;

#define MAX_ERROR_STRING_LENGTH 1000

ResultStatus WordRecognizer::KMeans(
      const IVectorsSet &_vectorsBase,
      const int               &_maxItersNum,
      const int               &_meansNum,
      const int               &_minVectorsNum,
      float*                        _means)
{
      int filesNum = _vectorsBase.GetSize();
      int vectorsSize = _vectorsBase.GetDim();

      if (filesNum <= 0) throw exception("KMeans: помилкові параметри
(_vectorsBase.GetSize() <= 0)");
      if (vectorsSize <= 0) throw exception("KMeans: помилкові параметри
(_vectorsBase.GetDim() <= 0)");
      if (_meansNum <= 0) throw exception("KMeans: помилкові параметри
(_meansNum <= 0)");
      if (_minVectorsNum <= 0) throw exception("KMeans: помилкові параметри
(_minVectorsNum <= 0)");
      if (_maxItersNum <= 0) throw exception("KMeans: помилкові параметри
(_maxItersNum <= 0)");

      int vectorsNum = 0;
      const float* pVector = NULL;
      float* pMean = NULL;
      int allVectorsNum = 0;

      int n, i, k, m, iter;
    float dist;
    float delta;
    float minDist;
    int nearestM;
      // Визначення мінімального і максимального значень всіх векторів для
початкової ініціалізації
      vector<float> minValData(vectorsSize);
      vector<float> maxValData(vectorsSize);
      float* minVal = &minValData[0];
      float* maxVal = &maxValData[0];

      for (k = 0; k < vectorsSize; k++)
      {
            minVal[k] = FLT_MAX;
            maxVal[k] = -FLT_MAX;
      }
      allVectorsNum = 0;
      for (n = 0; n < filesNum; n++)
      {
            _vectorsBase.GetVectors(n, vectorsNum, pVector);
            for (i = 0; i < vectorsNum; i++, pVector += vectorsSize)
            {
                  for (k = 0; k < vectorsSize; k++)
                  {
```

```
                        if (pVector[k] < minVal[k]) minVal[k] = pVector[k];
                        if (pVector[k] > maxVal[k]) maxVal[k] = pVector[k];
                  }
            }
            allVectorsNum += vectorsNum;
      }
      if (allVectorsNum < _minVectorsNum * _meansNum)
            throw exception("KMeans: недостатньо даних (числок всіх векторів <
(min кількість векторів) * (кількість середніх))");

      vector<int> vectorsNumInClusterData(_meansNum);
      vector<float> newMeansData(_meansNum * vectorsSize);
      int* vectorsNumInCluster = &vectorsNumInClusterData[0];
      float* newMeans = &newMeansData[0];
      // Початкова ініціалізація
    srand(0);
      pMean = _means;
    for (m = 0; m < _meansNum; m++, pMean += vectorsSize)
    {
        for (k = 0; k < vectorsSize; k++)
        {
            pMean[k] = minVal[k] + (maxVal[k] - minVal[k]) * (float)rand() /
(float)RAND_MAX;
        }
    }

    for (iter = 0; iter < _maxItersNum; iter++)
    {

        memset(vectorsNumInCluster, 0, sizeof(int) * _meansNum);
        memset(newMeans, 0, sizeof(float) * _meansNum * vectorsSize);

            for (n = 0; n < filesNum; n++)
            {
                  _vectorsBase.GetVectors(n, vectorsNum, pVector);
                  for (i = 0; i < vectorsNum; i++, pVector += vectorsSize)
                  {
                        // Знаходимо найближче середнє значення для чергового
вектора pVector
                        minDist = FLT_MAX;
                        nearestM = -1;
                        pMean = _means;
                        for (m = 0; m < _meansNum; m++, pMean += vectorsSize)
                        {
                              dist = 0;
                              for (k = 0; k < vectorsSize; k++)
                              {
                                    delta = pMean[k] - pVector[k];
                                    dist += delta * delta;
                              }
                              dist = sqrt(dist);

                              if (dist < minDist)
                              {
                                    minDist = dist;
                                    nearestM = m;
                              }
                        }
                        if (nearestM < 0)
                              throw exception("KMeans: внутрішня помилка");

                        // Підсумовуємо для подальшого знаходження нового
середнього значення
                        pMean = newMeans + nearestM * vectorsSize;
                        for (k = 0; k < vectorsSize; k++)
                        {
                              pMean[k] += pVector[k];
                        }
                        vectorsNumInCluster[nearestM]++;
```

```
                }
            }

            pMean = newMeans;
            for (m = 0; m < _meansNum; m++, pMean += vectorsSize)
            {
                if (vectorsNumInCluster[m] >= _minVectorsNum)
                {
                    for (k = 0; k < vectorsSize; k++)
                    {
                        pMean[k] /= (float)vectorsNumInCluster[m];
                    }
                }
                else
                {
                    for (k = 0; k < vectorsSize; k++)
                    {
                        pMean[k] = minVal[k] + (maxVal[k] - minVal[k]) *
(float)rand() / (float)RAND_MAX;
                    }
                }
            }

            delta = 0;
            for (k = 0; k < _meansNum * vectorsSize; k++)
            {
                if (fabs(_means[k] - newMeans[k]) > delta) delta =
fabs(_means[k] - newMeans[k]);
            }
            memcpy(_means, newMeans, sizeof(float) * _meansNum * vectorsSize);
            if (delta < 2.f * FLT_MIN)
                return RS_SUCCESS;
    }
    return RS_MAX_ITERATION_ACHIEVED;
}

void WordRecognizer::Adaptation(
      const int         &_vectorsSize,
      const int         &_vectorsNum,
      const int         &_meansNum,
      const float*      _means,
      const float       &_weight,
      const float*      _vectors,
      float*                  _result)
{
      if (_weight < 0.f) throw exception("Adaptation: помилкові параметри
(_weight < 0)");
      if (_vectorsSize <= 0) throw exception("Adaptation: помилкові параметри
(_vectorsSize < 0)");
      if (_vectorsNum < 0) throw exception("Adaptation: помилкові параметри
(_vectorsNum < 0)");
      if (_meansNum <= 0) throw exception("Adaptation: помилкові параметри
(_meansNum < 0)");

      const float* pVector = NULL;
      const float* pMean = NULL;
      float* pResult = NULL;
      float minD = FLT_MAX;
      float d = 0;
      float x = 0;
      int minJ = 0;
      int i, j, k;

      vector<int> n(_meansNum);
      memset(&n[0], 0, sizeof(int) * _meansNum);
      memset(_result, 0, sizeof(float) * _meansNum * _vectorsSize);

      pVector = _vectors;
      for (i = 0; i < _vectorsNum; i++, pVector += _vectorsSize)
```

```
	{
		minD = FLT_MAX;
		minJ = 0;
		d = 0;
		pMean = _means;
		for (j = 0; j < _meansNum; j++, pMean += _vectorsSize)
		{
			d = 0;
			for (k = 0; k < _vectorsSize; k++)
			{
				x = (pVector[k] - pMean[k]);
				d += x * x;
			}
			d = sqrt(d);
			if (d < minD)
			{
				minD = d;
				minJ = j;
			}
		}
		n[minJ]++;
		pResult = _result + minJ * _vectorsSize;
		for (k = 0; k < _vectorsSize; k++)
			pResult[k] += pVector[k];
	}
	pMean = _means;
	pResult = _result;
	for (j = 0; j < _meansNum; j++, pMean += _vectorsSize, pResult +=
_vectorsSize)
	{
		if (n[j] > 0)
		{
			d = 1.f / (_weight + (float)n[j]);
			for (k = 0; k < _vectorsSize; k++)
				pResult[k] = (_weight * pMean[k] + pResult[k]) * d;
		}
		else
		{
			memcpy(pResult, pMean, sizeof(float) * _vectorsSize);
		}
	}
}

ResultStatus WordRecognizer::CreateLDAMatrix(
	const IVectorsSet &_vectorsBase,
	const int		&_rows,
	float*			_matrix,
	float*			_meanVector
	)
{
	if (_rows <= 0) throw exception("CreateLDAMatrix: помилкові параметри
(_rows <= 0)");
	if (_vectorsBase.GetSize() <= 1) throw exception("CreateLDAMatrix:
помилкові параметри (_vectorsBase.GetSize() <= 0)");
	int vectorsSize = _vectorsBase.GetDim();
	int allVectorsNum = 0;
	int vectorsNum = 0;
	const float* pVector = NULL;
	for (int n = 0; n < _vectorsBase.GetSize(); n++)
	{
		_vectorsBase.GetVectors(n, vectorsNum, pVector);
		if (vectorsNum <= 0)
			ThrowFormatException("CreateLDAMatrix: число векторів для
класу %d <= 0", n);
		allVectorsNum += vectorsNum;
	}

	memset(_meanVector, 0, sizeof(float) * vectorsSize);
	// Знаходження середнього значення
```

```
	for (int n = 0; n < _vectorsBase.GetSize(); n++)
	{
		_vectorsBase.GetVectors(n, vectorsNum, pVector);
		for (int i = 0; i < vectorsNum; i++)
		{

			for (int k = 0; k < vectorsSize; k++)
			{
				_meanVector[k] += pVector[k];
			}
		}
	}
	for (int k = 0; k < vectorsSize; k++)
		_meanVector[k] /= (float)allVectorsNum;

	vector<double> contentData(allVectorsNum * (vectorsSize + 1));
	double* pContentData = &contentData[0];

	// Копіювання даних для LDA
	for (int n = 0; n < _vectorsBase.GetSize(); n++)
	{
		_vectorsBase.GetVectors(n, vectorsNum, pVector);
		for (int i = 0; i < vectorsNum; i++, pContentData += vectorsSize +
1)
		{

			for (int k = 0; k < vectorsSize; k++)
			{
				pContentData[k] = (double)pVector[k] -
(double)_meanVector[k];
			}
			pContentData[vectorsSize] = (double)n;
		}
	}

	alglib::real_2d_array data;
	alglib::real_2d_array result;
	data.setcontent(allVectorsNum, (vectorsSize + 1), &contentData[0]);

	alglib::ae_int_t info;

	alglib::fisherldan(data, allVectorsNum, vectorsSize,
_vectorsBase.GetSize(), info, result);

	if (info > 0)
	{
		for (int i = 0; i < _rows; i++)
		{
			for (int k = 0; k < vectorsSize; k++)
			{
				//_matrix[i * vectorsSize + k] = (float)result[i][k];
				_matrix[i * vectorsSize + k] = (float)result[k][i];
			}
		}
		if (info == 1)
			return RS_SUCCESS;
		else if (info == 2)
			return RS_MULTICOLLINEARITY;
		return RS_UNKNOWN_RESULT;
	}
	else
	{
		ThrowFormatException("CreateLDAMatrix: помилка alblib (%d)", info);
	}
	return RS_SUCCESS;
}

void WordRecognizer::MakeProjection(
	const int		&_rows,
```

```
	const int		&_vectorSize,
	const float*		_matrix,
	const float*		_meanVector,
	const float*		_vector,
	float*			_resultVector)
{
	if (_rows <= 0) throw exception("MakeProjection: помилкові параметри
(_rows <= 0)");
	if (_vectorSize <= 0) throw exception("MakeProjection: помилкові параметри
(_vectorSize <= 0)");

	float sum;
	int i, j;
	const float* pRow = _matrix;
	for (i = 0; i < _rows; i++, pRow += _vectorSize)
	{
		sum = 0;
		for (j = 0; j < _vectorSize; j++)
			sum += pRow[j] * (_vector[j] - _meanVector[j]);
		_resultVector[i] = sum;
	}
}

ResultStatus WordRecognizer::CreateCompareSystem(
	const IVectorsSet &_vectorsBase,
	float*			_meanBase,
	float*			_sigmaBase,
	const float		&_minSigma)
{
	int meansNum = _vectorsBase.GetSize();
	int vectorsSize = _vectorsBase.GetDim();

	if (meansNum <= 0) throw exception("CreateCompareSystem: помилкові
параметри (_vectorsBase.GetSize() <= 0)");
	if (vectorsSize <= 0) throw exception("CreateCompareSystem: помилкові
параметри (_vectorsBase.GetDim() <= 0)");
	if (_minSigma <= 0.f) throw exception("CreateCompareSystem: помилкові
параметри (_minSigma <= 0)");

	ResultStatus result = RS_SUCCESS;

	float d = 0;
	float x = 0;
	float sigma = 0;
	float minD = FLT_MAX;
	int vectorsNum = 0;
	int i, j, k;
	const float* pVector = NULL;
	float* pMean = NULL;

	memset(_meanBase, 0, sizeof(float) * meansNum * vectorsSize);
	memset(_sigmaBase, 0, sizeof(float) * meansNum);

	pMean = _meanBase;
	for (i = 0; i < meansNum; i++, pMean += vectorsSize)
	{
		_vectorsBase.GetVectors(i, vectorsNum, pVector);
		if (vectorsNum <= 0)
			throw exception("CreateCompareSystem: число векторів для класу
%d <= 0", i);

		sigma = 0;
		for (j = 0; j < vectorsNum; j++, pVector += vectorsSize)
		{
			for (k = 0; k < vectorsSize; k++)
			{
				x = pVector[k];
				pMean[k] += x;
				sigma += x * x;
```

```
                }
            }

            x = 1.f/(float)vectorsNum;
            d = 0;
            for (k = 0; k < vectorsSize; k++)
            {
                pMean[k] *= x;
                d += pMean[k] * pMean[k];
            }

            _sigmaBase[i] = sqrt(fabs(sigma * x - d));
            if (_sigmaBase[i] < _minSigma)
            {
                result = RS_MIN_SIGMA_USED;
                _sigmaBase[i] = _minSigma;
            }
        }
        return result;
}

void WordRecognizer::GetProbability(
        const int         &_vectorsSize,
        const int         &_meansNum,
        const float*      _meanBase,
        const float*      _sigmaBase,
        const float*      _testVector,
        bool              _useSigma,
        float*                  _result
        )
{
        if (_vectorsSize <= 0) throw exception("GetProbability: помилкові
параметри (_vectorsSize <= 0)");
        if (_meansNum <= 0) throw exception("GetProbability: помилкові параметри
(_meansNum <= 0)");

        int i, k;
        float d = 0;
        float x = 0;
        float sum = 0;
        float minD = 1.0e-10f;
        const float* pMean = _meanBase;

        for (i = 0; i < _meansNum; i++, pMean += _vectorsSize)
        {
            d = 0;
            for (k = 0; k < _vectorsSize; k++)
            {
                x = pMean[k] - _testVector[k];
                d += x * x;
            }
            if (d < minD) d = minD;
            _result[i] = (_useSigma) ? _sigmaBase[i] / sqrt(d) : 1.f / sqrt(d);
            sum += _result[i];
        }
        for (i = 0; i < _meansNum; i++)
            _result[i] /= sum;
}

void WordRecognizer::ThrowFormatException(const char* _msg, ...)
{
        char str[MAX_ERROR_STRING_LENGTH];
        va_list args;
        va_start(args, _msg);
#ifdef WIN32
        vsprintf_s<MAX_ERROR_STRING_LENGTH>(str, _msg, args);
#else
        vsprintf(str, _errorMsg, args);
#endif
```

```
	va_end(args);
	throw exception(str);
}

const char* WordRecognizer::GetMessageFromStatus(const ResultStatus &_status)
{
	switch(_status)
	{
	case RS_SUCCESS:
		return "success";
	case RS_MAX_ITERATION_ACHIEVED:
		return "maximum iteration achieved";
	case RS_MIN_SIGMA_USED:
		return "min sigma was used";
	case RS_MULTICOLLINEARITY:
		return "multicollinearity in training set";
	default:
		return "unknown status";
	}
}
```

// WordRecognizer.h – заголовний файл для модуля розпізнавання голосових команд

```
#ifndef WORD_RECOGNIZER__H
#define WORD_RECOGNIZER__H

//! Набір функціоналу для побудови системи розпізнавання слів по короткому
словнику
/*!
	Навчання системи:
	- Знаходження середніх векторів на всій базі навчання (KMeans)
	- Адаптація середніх векторів для кожного файлу з бази навчання
(Adaptation)
	- Побудова LDA-матриці для адаптованих середніх векторів (CreateLDAMatrix)

	- Знаходження проекцій адаптованих середніх векторів на площину LDA-
матриці	(MakeProjection)
	- Знаходження середніх значень і СКО знайдених проекцій усередині кожного
класу (CreateCompareSystem)


	Використання системи:

	- Для тестованої фрази проводиться адаптація середніх векторів
(Adaptation)
	- Адаптовані середні вектори проектуються на площину матриці LDA
(MakeProjection)
	- Для отриманої проекції знаходиться ймовірність приналежності до кожного
класу (GetProbability)
*/
namespace WordRecognizer
{
	//! Статус виконання завдання
	enum ResultStatus
	{
		RS_SUCCESS                      = 0,
		RS_MAX_ITERATION_ACHIEVED	= 1,
		RS_MIN_SIGMA_USED               = 2,
		RS_MULTICOLLINEARITY            = 3,
		RS_UNKNOWN_RESULT               = 4
	};

	//! Абстрактний клас для роботи з файлами (наборами, класами) векторів
	class IVectorsSet
	{
	public:

		//! \return розмір векторів
		virtual const int GetDim() const = 0;

		//! \return число файлів (наборів) векторів
		virtual const int GetSize() const = 0;

		//! Получение чергового файлу (набору) векторів
		/*!
			\param[in]  _indx		номер файлу (набору)
			\param[out] _vectorsNum		число векторів у файлі (наборі)
			\param[out] _vectors		покажчик на масив [_vectorsNum *
GetDim()] векторов
		*/
		virtual void GetVectors(
			const int   &_indx,
			int		&_vectorsNum,
			float*		&_vectors) = 0;

		virtual void GetVectors(
			const int	&_indx,
			int			&_vectorsNum,
			const float*	&_vectors) const = 0;
```

```
	};

	//! Побудова середніх векторів (метод K- середніх)
	/*!
		\param[in]  _vectorsBase	база всіх векторів (всі файли векторів
ознак бази навчання)
		\param[in]  _maxItersNum	максимальне число ітерацій
		\param[in]  _meansNum		число середніх векторів
		\param[in]  _minVectorsNum	мінімальне число векторів, які
відповідають одному середньому вектору
		\param[out] _means			масив [_vectorsBase.GetDim() *
_meansNum], що містить значення середніх векторів
	*/
	ResultStatus KMeans(
		const IVectorsSet &_vectorsBase,
		const int		&_maxItersNum,
		const int		&_meansNum,
		const int		&_minVectorsNum,
		float*			_means);

	//! Адаптація середніх векторів новими даними
	/*!
		\param[in]  _vectorsSize	розмір векторів
		\param[in]  _vectorsNum		число векторів нових даних
		\param[in]  _meansNum		число середніх векторів
		\param[in]  _means			масив [_vectorsSize * _meansNum]
середніх векторів
		\param[in]  _weight			емпіричне значення "ваги" нових
даних по відношенню до середніх векторів (>= 0)
		\param[in]  _vectors		масив [_vectorsSize * _vectorsNum]
векторів нових даних
		\param[out] _result			масив [_vectorsSize * _meansNum],
що містить значення адаптованих середніх векторів
	*/
	void Adaptation(
		const int	&_vectorsSize,
		const int	&_vectorsNum,
		const int	&_meansNum,
		const float*	_means,
		const float	&_weight,
		const float*	_vectors,
		float*		_result);

	//! Створення LDA-матриці
	/*!
		\param[in]  _vectorsBase	набори векторів, розбиті на відповідні
класи
		\param[in]  _rows		число рядків LDA- матриці
		\param[out] _matrix		масив [_rows *
_vectorsBase.GetDim()],що містить LDA- матрицю
		\param[out] _meanVector		масив [_vectorsBase.GetDim()], що
містить середній вектор
		\attention			число класів має бути більше 1

	*/
	ResultStatus CreateLDAMatrix(
		const IVectorsSet &_vectorsBase,
		const int		&_rows,
		float*			_matrix,
		float*			_meanVector
		);

	//! Центрування вектора і множення його на матрицю: x' = L(x-m), де L -
матриця, m - середнє значення векторів
	/*!
		\param[in] _rows		число рядків матриці
		\param[in] _vectorSize		розмір вхідного вектора
		\param[in] _matrix		масив [_rows * _vectorSize], що
містить матрицю (L)
```

```
            \param[in] _meanVector       массив[_vectorSize] середнього
значення векторів
            \param[in] _vector                  масив [_vectorSize] вихідного
вектора (x)
            \param[out] _resultVector    масив [_rows], що містить результуючий
вектор (x')
      */
      void MakeProjection(
            const int         &_rows,
            const int         &_vectorSize,
            const float*      _matrix,
            const float*      _meanVector,
            const float*      _vector,
            float*                  _resultVector);

      //! Підрахунок середніх значень і СКО для векторів бази навчання
      /*!
            \param[in]  _vectorsBase     вектора бази навчання, розбиті по
класах
            \param[in]  _meansBase       масив [_vectorsBase.GetSize() *
_vectorsBase.GetDim()],що містить середні значення класів бази навчання
            \param[in]  _sigmaBase       масив [_vectorsBase.GetSize()], що
містить СКО усередині кожного класу бази навчання
            \param[in]  _minSigma        мінімальне значення для СКО
      */
      ResultStatus CreateCompareSystem(
            const IVectorsSet &_vectorsBase,
            float*                        _meanBase,
            float*                        _sigmaBase,
            const float             &_minSigma);

      //! Підрахунок ймовірності належності тестованого вектора до одного з
класів бази навчання
      /*!
            \param[in]  _vectorsSize     розмір векторів
            \param[in]  _meansNum        число середніх векторів класів бази
навчання
            \param[in]  _meanBase        масив [_meansNum * _vectorsSize]
середніх векторів класів бази навчання
            \param[in]  _sigmaBase       масив [_vectorsSize] СКО усередині
класів бази навчання
            \param[in]  _testVector      масив [_vectorsSize], що містить
тестований вектор
            \param[in]  _useSigma        використовувати нормування відстані на
СКО
            \param[out] _result                 масив [_meansNum], що містить
значення ймовірності (від 0 до 1) приналежності тестованого вектора відповідному
класу бази навчання
      */
      void GetProbability(
            const int         &_vectorsSize,
            const int         &_meansNum,
            const float*      _meanBase,
            const float*      _sigmaBase,
            const float*      _testVector,
            bool              _useSigma,
            float*                  _result
            );
      //! Виведення виключення std::exception з повідомленням у вказаному
форматі (аналогічно sprintf)
      void ThrowFormatException(const char* _msg, ...);

      //! Розшифровка статусу виконання задачі
      const char* GetMessageFromStatus(const ResultStatus &_status);
}
#endif
```

// diffequations.cpp – модуль обчислення диференціальних рівнянь, використовуваних у модулі розпізнавання голосових команд

```
#include "stdafx.h"
#include "diffequations.h"

// вимкнення деяких несуттєвих попереджень
#if (AE_COMPILER==AE_MSVC)
#pragma warning(disable:4100)
#pragma warning(disable:4127)
#pragma warning(disable:4702)
#pragma warning(disable:4996)
#endif
using namespace std;

// реалізація C++ інтерфейсу

namespace alglib
{


/*************************************************************************

*************************************************************************/
_odesolverstate_owner::_odesolverstate_owner()
{
    p_struct =
(alglib_impl::odesolverstate*)alglib_impl::ae_malloc(sizeof(alglib_impl::odesolv
erstate), NULL);
    if( p_struct==NULL )
        throw ap_error("ALGLIB: malloc error");
    if( !alglib_impl::_odesolverstate_init(p_struct, NULL, ae_false) )
        throw ap_error("ALGLIB: malloc error");
}

_odesolverstate_owner::_odesolverstate_owner(const _odesolverstate_owner &rhs)
{
    p_struct =
(alglib_impl::odesolverstate*)alglib_impl::ae_malloc(sizeof(alglib_impl::odesolv
erstate), NULL);
    if( p_struct==NULL )
        throw ap_error("ALGLIB: malloc error");
    if( !alglib_impl::_odesolverstate_init_copy(p_struct,
const_cast<alglib_impl::odesolverstate*>(rhs.p_struct), NULL, ae_false) )
        throw ap_error("ALGLIB: malloc error");
}

_odesolverstate_owner& _odesolverstate_owner::operator=(const
_odesolverstate_owner &rhs)
{
    if( this==&rhs )
        return *this;
    alglib_impl::_odesolverstate_clear(p_struct);
    if( !alglib_impl::_odesolverstate_init_copy(p_struct,
const_cast<alglib_impl::odesolverstate*>(rhs.p_struct), NULL, ae_false) )
        throw ap_error("ALGLIB: malloc error");
    return *this;
}

_odesolverstate_owner::~_odesolverstate_owner()
{
    alglib_impl::_odesolverstate_clear(p_struct);
    ae_free(p_struct);
}

alglib_impl::odesolverstate* _odesolverstate_owner::c_ptr()
{
    return p_struct;
```

```
}

alglib_impl::odesolverstate* _odesolverstate_owner::c_ptr() const
{
    return const_cast<alglib_impl::odesolverstate*>(p_struct);
}
odesolverstate::odesolverstate() : _odesolverstate_owner() ,needdy(p_struct-
>needdy),y(&p_struct->y),dy(&p_struct->dy),x(p_struct->x)
{
}

odesolverstate::odesolverstate(const odesolverstate
&rhs):_odesolverstate_owner(rhs) ,needdy(p_struct->needdy),y(&p_struct-
>y),dy(&p_struct->dy),x(p_struct->x)
{
}

odesolverstate& odesolverstate::operator=(const odesolverstate &rhs)
{
    if( this==&rhs )
        return *this;
    _odesolverstate_owner::operator=(rhs);
    return *this;
}

odesolverstate::~odesolverstate()
{
}


/*************************************************************************

*************************************************************************/
_odesolverreport_owner::_odesolverreport_owner()
{
    p_struct =
(alglib_impl::odesolverreport*)alglib_impl::ae_malloc(sizeof(alglib_impl::odesol
verreport), NULL);
    if( p_struct==NULL )
        throw ap_error("ALGLIB: malloc error");
    if( !alglib_impl::_odesolverreport_init(p_struct, NULL, ae_false) )
        throw ap_error("ALGLIB: malloc error");
}

_odesolverreport_owner::_odesolverreport_owner(const _odesolverreport_owner
&rhs)
{
    p_struct =
(alglib_impl::odesolverreport*)alglib_impl::ae_malloc(sizeof(alglib_impl::odesol
verreport), NULL);
    if( p_struct==NULL )
        throw ap_error("ALGLIB: malloc error");
    if( !alglib_impl::_odesolverreport_init_copy(p_struct,
const_cast<alglib_impl::odesolverreport*>(rhs.p_struct), NULL, ae_false) )
        throw ap_error("ALGLIB: malloc error");
}

_odesolverreport_owner& _odesolverreport_owner::operator=(const
_odesolverreport_owner &rhs)
{
    if( this==&rhs )
        return *this;
    alglib_impl::_odesolverreport_clear(p_struct);
    if( !alglib_impl::_odesolverreport_init_copy(p_struct,
const_cast<alglib_impl::odesolverreport*>(rhs.p_struct), NULL, ae_false) )
        throw ap_error("ALGLIB: malloc error");
    return *this;
}
```

```
_odesolverreport_owner::~_odesolverreport_owner()
{
    alglib_impl::_odesolverreport_clear(p_struct);
    ae_free(p_struct);
}

alglib_impl::odesolverreport* _odesolverreport_owner::c_ptr()
{
    return p_struct;
}

alglib_impl::odesolverreport* _odesolverreport_owner::c_ptr() const
{
    return const_cast<alglib_impl::odesolverreport*>(p_struct);
}
odesolverreport::odesolverreport() : _odesolverreport_owner() ,nfev(p_struct-
>nfev),terminationtype(p_struct->terminationtype)
{
}

odesolverreport::odesolverreport(const odesolverreport
&rhs):_odesolverreport_owner(rhs) ,nfev(p_struct-
>nfev),terminationtype(p_struct->terminationtype)
{
}

odesolverreport& odesolverreport::operator=(const odesolverreport &rhs)
{
    if( this==&rhs )
        return *this;
    _odesolverreport_owner::operator=(rhs);
    return *this;
}

odesolverreport::~odesolverreport()
{
}

/*************************************************************************

Розв'язання диф.рівняння Y'=f(Y,x) із початковими умовами  Y(xs)=Ys
(Y може бути скалярною змінною або вектором N змінних).

ВХІДНІ ПАРАМЕТРИ:
    Y       -   початкові умови, масив[0..N-1].
                містить значення Y[] при X[0]
    N       -   розмір системи
    X       -   точки, в яких Y табулюється, масив[0..M-1]
                інтегрування починається в X[0], закінчується в X[M-1],
                проміжні значення при X[i] повертаються також.
                МАЮТЬ БУТИ ВПОРЯДКОВАНІ ЗА ЗРОСТАННЯМ, АБО ЗА СПАДАННЯМ!!!
    M       -   число проміжних точок + перша + остання точки:
                * M>2 означає, що необхідні як Y(X[M-1]), так і M-2 значень
                  проміжних точок
                * M=2 означає, що слід інтегрувати від X[0] до X[1],
                  Не цікавлячись проміжними значеннями.
                * M=1 означає, що ви не хочете інтегрувати
                  це вироджений випадок, але він обробляється правильно.
                * M<1 означає помилку
    Eps     -   допустиме відхилення (абсолютна/відносна похибка
                на кожному кроці менше Eps.
                * Eps>0, розглядається абсолютна похибка
                * Eps<0, розглядається відносна похибка



    H       -   початкова довжина кроку, налаштовується автоматично
                після першого кроку. Якщо H=0, крок буде вибрано
                автоматично (звичайно він буде дорівнювати 0.001 від
```

```
                min(x[i]-x[j])).

ВИХІДНІ ПАРАМЕТРИ
    State   -   структура, яка зберігає стан алгоритму між послідовними
                викликами OdeSolverIteration. Використовується для зворотнього
                Зв'язку. Ця структура має слідувати за підпрограмою
                OdeSolverIteration.




*************************************************************************/
void odesolverrkck(const real_1d_array &y, const ae_int_t n, const real_1d_array
&x, const ae_int_t m, const double eps, const double h, odesolverstate &state)
{
    alglib_impl::ae_state _alglib_env_state;
    alglib_impl::ae_state_init(&_alglib_env_state);
    try
    {

alglib_impl::odesolverrkck(const_cast<alglib_impl::ae_vector*>(y.c_ptr()), n,
const_cast<alglib_impl::ae_vector*>(x.c_ptr()), m, eps, h,
const_cast<alglib_impl::odesolverstate*>(state.c_ptr()), &_alglib_env_state);
        alglib_impl::ae_state_clear(&_alglib_env_state);
        return;
    }
    catch(alglib_impl::ae_error_type)
    {
        throw ap_error(_alglib_env_state.error_msg);
    }
    catch(...)
    {
        throw;
    }
}

/*************************************************************************


Розв'язання диф.рівняння Y'=f(Y,x) із початковими умовами  Y(xs)=Ys
(Y може бути скалярною змінною або вектором N змінних).

ВХІДНІ ПАРАМЕТРИ:
    Y       -   початкові умови, масив[0..N-1].
                містить значення Y[] при X[0]
    N       -   розмір системи
    X       -   точки, в яких Y табулюється, масив[0..M-1]
                інтегрування починається в X[0], закінчується в X[M-1],
                проміжні значення при X[i] повертаються також.
                МАЮТЬ БУТИ ВПОРЯДКОВАНІ ЗА ЗРОСТАННЯМ, АБО ЗА СПАДАННЯМ!!!
    M       -   число проміжних точок + перша + остання точки:
                * M>2 означає, що необхідні як Y(X[M-1]), так і M-2 значень
                  проміжних точок
                * M=2 означає, що слід інтегрувати від X[0] до X[1],
                  Не цікавлячись проміжними значеннями.
                * M=1 означає, що ви не хочете інтегрувати
                  це вироджений випадок, але він обробляється правильно.
                * M<1 означає помилку
    Eps     -   допустиме відхилення (абсолютна/відносна похибка
                на кожному кроці менше Eps.
                * Eps>0, розглядається абсолютна похибка
                * Eps<0, розглядається відносна похибка



    H       -   початкова довжина кроку, налаштовується автоматично
                після першого кроку. Якщо H=0, крок буде вибрано
                автоматично (звичайно він буде дорівнювати 0.001 від
```

```
                min(x[i]-x[j])).

ВИХІДНІ ПАРАМЕТРИ
    State   -   структура, яка зберігає стан алгоритму між послідовними
                викликами OdeSolverIteration. Використовується для зворотнього
                Зв'язку. Ця структура має слідувати за підпрограмою
                OdeSolverIteration.




*************************************************************************/
void odesolverrkck(const real_1d_array &y, const real_1d_array &x, const double
eps, const double h, odesolverstate &state)
{
    alglib_impl::ae_state _alglib_env_state;
    ae_int_t n;
    ae_int_t m;

    n = y.length();
    m = x.length();
    alglib_impl::ae_state_init(&_alglib_env_state);
    try
    {

alglib_impl::odesolverrkck(const_cast<alglib_impl::ae_vector*>(y.c_ptr()), n,
const_cast<alglib_impl::ae_vector*>(x.c_ptr()), m, eps, h,
const_cast<alglib_impl::odesolverstate*>(state.c_ptr()), &_alglib_env_state);

        alglib_impl::ae_state_clear(&_alglib_env_state);
        return;
    }
    catch(alglib_impl::ae_error_type)
    {
        throw ap_error(_alglib_env_state.error_msg);
    }
    catch(...)
    {
        throw;
    }
}

/*************************************************************************
інтерфейс зворотнього зв'язку
*************************************************************************/
bool odesolveriteration(const odesolverstate &state)
{
    alglib_impl::ae_state _alglib_env_state;
    alglib_impl::ae_state_init(&_alglib_env_state);
    try
    {
        ae_bool result =
alglib_impl::odesolveriteration(const_cast<alglib_impl::odesolverstate*>(state.c
_ptr()), &_alglib_env_state);
        alglib_impl::ae_state_clear(&_alglib_env_state);
        return *(reinterpret_cast<bool*>(&result));
    }
    catch(alglib_impl::ae_error_type)
    {
        throw ap_error(_alglib_env_state.error_msg);
    }
    catch(...)
    {
        throw;
    }
}
```

```
void odesolversolve(odesolverstate &state,
    void (*diff)(const real_1d_array &y, double x, real_1d_array &dy, void
*ptr),
    void *ptr){
    alglib_impl::ae_state _alglib_env_state;
    if( diff==NULL )
        throw ap_error("ALGLIB: error in 'odesolversolve()' (diff is NULL)");
    alglib_impl::ae_state_init(&_alglib_env_state);
    try
    {
        while( alglib_impl::odesolveriteration(state.c_ptr(),
&_alglib_env_state) )
        {
            if( state.needdy )
            {
                diff(state.y, state.x, state.dy, ptr);
                continue;
            }
            throw ap_error("ALGLIB: unexpected error in 'odesolversolve'");
        }
        alglib_impl::ae_state_clear(&_alglib_env_state);
    }
    catch(alglib_impl::ae_error_type)
    {
        throw ap_error(_alglib_env_state.error_msg);
    }
    catch(...)
    {
        throw;
    }
}



/*************************************************************************
Результати роботи ODE solver



ВХІДНІ ПАРАМЕТРИ:
    State   -   стан алгоритму (використовується в OdeSolverIteration).

ВИХІДНІ ПАРАМЕТРИ:
    M       -   кількість табульованих значень, M>=1
    XTbl    -   масив[0..M-1], значення X
    YTbl    -   масив[0..M-1,0..N-1], значення Y в X[i]
    Rep     -   звіт:
                * Rep.TerminationType завершення коду:
                    * -2    X не впорядкований по зростанню/спаданню або
                            є однакові значення X[], тобто X[i]=X[i+1]
                    * -1    задані неправильні параметри
                    *  1    задача розв'язана
                * Rep.NFEV містить кількість обчислень функції

*************************************************************************/
void odesolverresults(const odesolverstate &state, ae_int_t &m, real_1d_array
&xtbl, real_2d_array &ytbl, odesolverreport &rep)
{
    alglib_impl::ae_state _alglib_env_state;
    alglib_impl::ae_state_init(&_alglib_env_state);
    try
    {

alglib_impl::odesolverresults(const_cast<alglib_impl::odesolverstate*>(state.c_p
tr()), &m, const_cast<alglib_impl::ae_vector*>(xtbl.c_ptr()),
const_cast<alglib_impl::ae_matrix*>(ytbl.c_ptr()),
const_cast<alglib_impl::odesolverreport*>(rep.c_ptr()), &_alglib_env_state);
        alglib_impl::ae_state_clear(&_alglib_env_state);
        return;
```

```
    }
    catch(alglib_impl::ae_error_type)
    {
        throw ap_error(_alglib_env_state.error_msg);
    }
    catch(...)
    {
        throw;
    }
}
}

// імплементація обчислювального ядра

namespace alglib_impl
{
static double odesolver_odesolvermaxgrow = 3.0;
static double odesolver_odesolvermaxshrink = 10.0;
static void odesolver_odesolverinit(ae_int_t solvertype,
     /* Real    */ ae_vector* y,
     ae_int_t n,
     /* Real    */ ae_vector* x,
     ae_int_t m,
     double eps,
     double h,
     odesolverstate* state,
     ae_state *_state);




/*************************************************************************


Розв'язання диф.рівняння Y'=f(Y,x) із початковими умовами  Y(xs)=Ys
(Y може бути скалярною змінною або вектором N змінних).

ВХІДНІ ПАРАМЕТРИ:
    Y       -   початкові умови, масив[0..N-1].
                містить значення Y[] при X[0]
    N       -   розмір системи
    X       -   точки, в яких Y табулюється, масив[0..M-1]
                інтегрування починається в X[0], закінчується в X[M-1],
                проміжні значення при X[i] повертаються також.
                МАЮТЬ БУТИ ВПОРЯДКОВАНІ ЗА ЗРОСТАННЯМ, АБО ЗА СПАДАННЯМ!!!
    M       -   число проміжних точок + перша + остання точки:
                * M>2 означає, що необхідні як Y(X[M-1]), так і M-2 значень
                  проміжних точок
                * M=2 означає, що слід інтегрувати від X[0] до X[1],
                  Не цікавлячись проміжними значеннями.
                * M=1 означає, що ви не хочете інтегрувати
                  це вироджений випадок, але він обробляється правильно.
                * M<1 означає помилку
    Eps     -   допустиме відхилення (абсолютна/відносна похибка
                на кожному кроці менше Eps.
                * Eps>0, розглядається абсолютна похибка
                * Eps<0, розглядається відносна похибка



    H       -   початкова довжина кроку, налаштовується автоматично
                після першого кроку. Якщо H=0, крок буде вибрано
                автоматично (звичайно він буде дорівнювати 0.001 від
                min(x[i]-x[j])).

ВИХІДНІ ПАРАМЕТРИ
    State   -   структура, яка зберігає стан алгоритму між послідовними
                викликами OdeSolverIteration. Використовується для зворотнього
```

Зв’язку. Ця структура має слідувати за підпрограмою OdeSolverIteration.

```
*************************************************************************/
void odesolverrkck(/* Real    */ ae_vector* y,
     ae_int_t n,
     /* Real    */ ae_vector* x,
     ae_int_t m,
     double eps,
     double h,
     odesolverstate* state,
     ae_state *_state)
{

    _odesolverstate_clear(state);

    ae_assert(n>=1, "ODESolverRKCK: N<1!", _state);
    ae_assert(m>=1, "ODESolverRKCK: M<1!", _state);
    ae_assert(y->cnt>=n, "ODESolverRKCK: Length(Y)<N!", _state);
    ae_assert(x->cnt>=m, "ODESolverRKCK: Length(X)<M!", _state);
    ae_assert(isfinitevector(y, n, _state), "ODESolverRKCK: Y містить
нескінченні або NaN значення", _state);
    ae_assert(isfinitevector(x, m, _state), "ODESolverRKCK: Y містить
нескінченні або NaN значення", _state);
    ae_assert(ae_isfinite(eps, _state), "ODESolverRKCK: Eps є нескінченним!",
_state);
    ae_assert(ae_fp_neq(eps,0), "ODESolverRKCK: Eps дорівнює нулю!", _state);
    ae_assert(ae_isfinite(h, _state), "ODESolverRKCK: H є нескінченним!",
_state);
    odesolver_odesolverinit(0, y, n, x, m, eps, h, state, _state);
}


ae_bool odesolveriteration(odesolverstate* state, ae_state *_state)
{
    ae_int_t n;
    ae_int_t m;
    ae_int_t i;
    ae_int_t j;
    ae_int_t k;
    double xc;
    double v;
    double h;
    double h2;
    ae_bool gridpoint;
    double err;
    double maxgrowpow;
    ae_int_t klimit;
    ae_bool result;



    /*
     * Підготовка до зворотнього зв’язку
     */
    if( state->rstate.stage>=0 )
    {
        n = state->rstate.ia.ptr.p_int[0];
        m = state->rstate.ia.ptr.p_int[1];
        i = state->rstate.ia.ptr.p_int[2];
        j = state->rstate.ia.ptr.p_int[3];
        k = state->rstate.ia.ptr.p_int[4];
        klimit = state->rstate.ia.ptr.p_int[5];
        gridpoint = state->rstate.ba.ptr.p_bool[0];
        xc = state->rstate.ra.ptr.p_double[0];
        v = state->rstate.ra.ptr.p_double[1];
        h = state->rstate.ra.ptr.p_double[2];
```

```
        h2 = state->rstate.ra.ptr.p_double[3];
        err = state->rstate.ra.ptr.p_double[4];
        maxgrowpow = state->rstate.ra.ptr.p_double[5];
    }
    else
    {
        n = -983;
        m = -989;
        i = -834;
        j = 900;
        k = -287;
        klimit = 364;
        gridpoint = ae_false;
        xc = -338;
        v = -686;
        h = 912;
        h2 = 585;
        err = 497;
        maxgrowpow = -271;
    }
    if( state->rstate.stage==0 )
    {
        goto lbl_0;
    }

    /*
     * Тіло програми
     */

    /*
     * підготовка
     */
    if( state->repterminationtype!=0 )
    {
        result = ae_false;
        return result;
    }
    n = state->n;
    m = state->m;
    h = state->h;
    maxgrowpow = ae_pow(odesolver_odesolvermaxgrow, 5, _state);
    state->repnfev = 0;

    /*
     * деякі попередні перевірки на внутрішні помилки
     * після яких ми припускаємо, що H>0 і M>1
     */
    ae_assert(ae_fp_greater(state->h,0), "ODESolver: внутрішня помилка",
_state);
    ae_assert(m>1, "ODESolverIteration: внутрішня помилка", _state);

    /*
     * вибір вирішувального пристрою
     */
    if( state->solvertype!=0 )
    {
        goto lbl_1;
    }

    /*
     * Підготовка таблиці коефіцієнтів
     * Перевірка на помилки
     */
    ae_vector_set_length(&state->rka, 6, _state);
    state->rka.ptr.p_double[0] = 0;
    state->rka.ptr.p_double[1] = (double)1/(double)5;
    state->rka.ptr.p_double[2] = (double)3/(double)10;
    state->rka.ptr.p_double[3] = (double)3/(double)5;
    state->rka.ptr.p_double[4] = 1;
```

```
    state->rka.ptr.p_double[5] = (double)7/(double)8;
    ae_matrix_set_length(&state->rkb, 6, 5, _state);
    state->rkb.ptr.pp_double[1][0] = (double)1/(double)5;
    state->rkb.ptr.pp_double[2][0] = (double)3/(double)40;
    state->rkb.ptr.pp_double[2][1] = (double)9/(double)40;
    state->rkb.ptr.pp_double[3][0] = (double)3/(double)10;
    state->rkb.ptr.pp_double[3][1] = -(double)9/(double)10;
    state->rkb.ptr.pp_double[3][2] = (double)6/(double)5;
    state->rkb.ptr.pp_double[4][0] = -(double)11/(double)54;
    state->rkb.ptr.pp_double[4][1] = (double)5/(double)2;
    state->rkb.ptr.pp_double[4][2] = -(double)70/(double)27;
    state->rkb.ptr.pp_double[4][3] = (double)35/(double)27;
    state->rkb.ptr.pp_double[5][0] = (double)1631/(double)55296;
    state->rkb.ptr.pp_double[5][1] = (double)175/(double)512;
    state->rkb.ptr.pp_double[5][2] = (double)575/(double)13824;
    state->rkb.ptr.pp_double[5][3] = (double)44275/(double)110592;
    state->rkb.ptr.pp_double[5][4] = (double)253/(double)4096;
    ae_vector_set_length(&state->rkc, 6, _state);
    state->rkc.ptr.p_double[0] = (double)37/(double)378;
    state->rkc.ptr.p_double[1] = 0;
    state->rkc.ptr.p_double[2] = (double)250/(double)621;
    state->rkc.ptr.p_double[3] = (double)125/(double)594;
    state->rkc.ptr.p_double[4] = 0;
    state->rkc.ptr.p_double[5] = (double)512/(double)1771;
    ae_vector_set_length(&state->rkcs, 6, _state);
    state->rkcs.ptr.p_double[0] = (double)2825/(double)27648;
    state->rkcs.ptr.p_double[1] = 0;
    state->rkcs.ptr.p_double[2] = (double)18575/(double)48384;
    state->rkcs.ptr.p_double[3] = (double)13525/(double)55296;
    state->rkcs.ptr.p_double[4] = (double)277/(double)14336;
    state->rkcs.ptr.p_double[5] = (double)1/(double)4;
    ae_matrix_set_length(&state->rkk, 6, n, _state);

    /*
     * Головний цикл складається із двох ітерацій:
     * * зовнішня, в якій ми йдемо від X[i-1] до X[i]
     * * внутрішня, в якій ми змінюємо всередині [X[i-1],X[i]]
     */
    ae_matrix_set_length(&state->ytbl, m, n, _state);
    ae_vector_set_length(&state->escale, n, _state);
    ae_vector_set_length(&state->yn, n, _state);
    ae_vector_set_length(&state->yns, n, _state);
    xc = state->xg.ptr.p_double[0];
    ae_v_move(&state->ytbl.ptr.pp_double[0][0], 1, &state->yc.ptr.p_double[0],
1, ae_v_len(0,n-1));
    for(j=0; j<=n-1; j++)
    {
        state->escale.ptr.p_double[j] = 0;
    }
    i = 1;
lbl_3:
    if( i>m-1 )
    {
        goto lbl_5;
    }

    /*
     * begin inner iteration
     */
lbl_6:
    if( ae_false )
    {
        goto lbl_7;
    }

    /*
     * зменшуємо крок, якщо потрібно (за межі правої границі).
     * визначаємо зберігати X чи hidetermine should we store X or not
     */
```

```
    if( ae_fp_greater_eq(xc+h,state->xg.ptr.p_double[i]) )
    {
        h = state->xg.ptr.p_double[i]-xc;
        gridpoint = ae_true;
    }
    else
    {
        gridpoint = ae_false;
    }

    /*
     * Оновлюємо максимум шкали помилок
     *
     * Ці максимуми ініціалізуються нулями ,
     * а потім оновлюються на кожній ітерації.
     */
    for(j=0; j<=n-1; j++)
    {
        state->escale.ptr.p_double[j] = ae_maxreal(state-
>escale.ptr.p_double[j], ae_fabs(state->yc.ptr.p_double[j], _state), _state);
    }

    /*
     * виконуємо за один крок наступні операції:
     * 1. обчислюємо всю інформацію, необхідну для виконання кроку
     * 2. оновлюємо максимуми шкали помилок, використовуючи значення/їх похідні,
     *    отримані протягом (1)
     *
     * Слід брати до уваги, що ми використовуємо масштабування X, щоб звести
     * задачу до форми, де x[0] < x[1] < ... < x[n-1]. Так що X замінюється на
     * x=xscale*t, і dy/dx=f(y,x) замінюється
     * на dy/dt=xscale*f(y,xscale*t).
     */
    ae_v_move(&state->yn.ptr.p_double[0], 1, &state->yc.ptr.p_double[0], 1,
ae_v_len(0,n-1));
    ae_v_move(&state->yns.ptr.p_double[0], 1, &state->yc.ptr.p_double[0], 1,
ae_v_len(0,n-1));
    k = 0;
lbl_8:
    if( k>5 )
    {
        goto lbl_10;
    }

    /*
     * підготовка даних до наступного оновлення YN/YNS
     */
    state->x = state->xscale*(xc+state->rka.ptr.p_double[k]*h);
    ae_v_move(&state->y.ptr.p_double[0], 1, &state->yc.ptr.p_double[0], 1,
ae_v_len(0,n-1));
    for(j=0; j<=k-1; j++)
    {
        v = state->rkb.ptr.pp_double[k][j];
        ae_v_addd(&state->y.ptr.p_double[0], 1, &state->rkk.ptr.pp_double[j][0],
1, ae_v_len(0,n-1), v);
    }
    state->needdy = ae_true;
    state->rstate.stage = 0;
    goto lbl_rcomm;
lbl_0:
    state->needdy = ae_false;
    state->repnfev = state->repnfev+1;
    v = h*state->xscale;
    ae_v_moved(&state->rkk.ptr.pp_double[k][0], 1, &state->dy.ptr.p_double[0],
1, ae_v_len(0,n-1), v);

    /*
     * оновлення YN/YNS
     */
```

```
    v = state->rkc.ptr.p_double[k];
    ae_v_addd(&state->yn.ptr.p_double[0], 1, &state->rkk.ptr.pp_double[k][0], 1,
ae_v_len(0,n-1), v);
    v = state->rkcs.ptr.p_double[k];
    ae_v_addd(&state->yns.ptr.p_double[0], 1, &state->rkk.ptr.pp_double[k][0],
1, ae_v_len(0,n-1), v);
    k = k+1;
    goto lbl_8;
lbl_10:

    /*
     * оцінка помилки
     */
    err = 0;
    for(j=0; j<=n-1; j++)
    {
        if( !state->fraceps )
        {

            /*
             * оцінка абсолютної помилки
             */
            err = ae_maxreal(err, ae_fabs(state->yn.ptr.p_double[j]-state-
>yns.ptr.p_double[j], _state), _state);
        }
        else
        {

            /*
             * оцінка відносної помилки
             */
            v = state->escale.ptr.p_double[j];
            if( ae_fp_eq(v,0) )
            {
                v = 1;
            }
            err = ae_maxreal(err, ae_fabs(state->yn.ptr.p_double[j]-state-
>yns.ptr.p_double[j], _state)/v, _state);
        }
    }

    /*
     * обчислення нового кроку, рестарт, якщо необхідно
     */
    if( ae_fp_less_eq(maxgrowpow*err,state->eps) )
    {
        h2 = odesolver_odesolvermaxgrow*h;
    }
    else
    {
        h2 = h*ae_pow(state->eps/err, 0.2, _state);
    }
    if( ae_fp_less(h2,h/odesolver_odesolvermaxshrink) )
    {
        h2 = h/odesolver_odesolvermaxshrink;
    }
    if( ae_fp_greater(err,state->eps) )
    {
        h = h2;
        goto lbl_6;
    }

    /*
     * продвинуте положення
     */
    xc = xc+h;
    ae_v_move(&state->yc.ptr.p_double[0], 1, &state->yn.ptr.p_double[0], 1,
ae_v_len(0,n-1));
```

```
    /*
     * оновлення H
     */
    h = h2;

    /*
     * вихід на точці сітки
     */
    if( gridpoint )
    {
        goto lbl_7;
    }
    goto lbl_6;
lbl_7:

    /*
     * збереження результату
     */
    ae_v_move(&state->ytbl.ptr.pp_double[i][0], 1, &state->yc.ptr.p_double[0],
1, ae_v_len(0,n-1));
    i = i+1;
    goto lbl_3;
lbl_5:
    state->repterminationtype = 1;
    result = ae_false;
    return result;
lbl_1:
    result = ae_false;
    return result;

    /*
     * збереження стану
     */
lbl_rcomm:
    result = ae_true;
    state->rstate.ia.ptr.p_int[0] = n;
    state->rstate.ia.ptr.p_int[1] = m;
    state->rstate.ia.ptr.p_int[2] = i;
    state->rstate.ia.ptr.p_int[3] = j;
    state->rstate.ia.ptr.p_int[4] = k;
    state->rstate.ia.ptr.p_int[5] = klimit;
    state->rstate.ba.ptr.p_bool[0] = gridpoint;
    state->rstate.ra.ptr.p_double[0] = xc;
    state->rstate.ra.ptr.p_double[1] = v;
    state->rstate.ra.ptr.p_double[2] = h;
    state->rstate.ra.ptr.p_double[3] = h2;
    state->rstate.ra.ptr.p_double[4] = err;
    state->rstate.ra.ptr.p_double[5] = maxgrowpow;
    return result;
}


/*************************************************************************
Результати роботи ODE solver



ВХІДНІ ПАРАМЕТРИ:
    State   -   стан алгоритму (використовується в OdeSolverIteration).

ВИХІДНІ ПАРАМЕТРИ:
    M       -   кількість табульованих значень, M>=1
    XTbl    -   масив[0..M-1], значення X
    YTbl    -   масив[0..M-1,0..N-1], значення Y в X[i]
    Rep     -   звіт:
                * Rep.TerminationType завершення коду:
                    * -2    X не впорядкований по зростанню/спаданню або
                            є однакові значення X[], тобто X[i]=X[i+1]
                    * -1    задані неправильні параметри
```

```
                    *  1    задача розв'язана
                * Rep.NFEV містить кількість обчислень функції

*************************************************************************/
void odesolverresults(odesolverstate* state,
     ae_int_t* m,
     /* Real    */ ae_vector* xtbl,
     /* Real    */ ae_matrix* ytbl,
     odesolverreport* rep,
     ae_state *_state)
{
    double v;
    ae_int_t i;

    *m = 0;
    ae_vector_clear(xtbl);
    ae_matrix_clear(ytbl);
    _odesolverreport_clear(rep);

    rep->terminationtype = state->repterminationtype;
    if( rep->terminationtype>0 )
    {
        *m = state->m;
        rep->nfev = state->repnfev;
        ae_vector_set_length(xtbl, state->m, _state);
        v = state->xscale;
        ae_v_moved(&xtbl->ptr.p_double[0], 1, &state->xg.ptr.p_double[0], 1,
ae_v_len(0,state->m-1), v);
        ae_matrix_set_length(ytbl, state->m, state->n, _state);
        for(i=0; i<=state->m-1; i++)
        {
            ae_v_move(&ytbl->ptr.pp_double[i][0], 1, &state-
>ytbl.ptr.pp_double[i][0], 1, ae_v_len(0,state->n-1));
        }
    }
    else
    {
        rep->nfev = 0;
    }
}


/*************************************************************************
Внутрішня ініціалізація підпрограми
*************************************************************************/
static void odesolver_odesolverinit(ae_int_t solvertype,
     /* Real    */ ae_vector* y,
     ae_int_t n,
     /* Real    */ ae_vector* x,
     ae_int_t m,
     double eps,
     double h,
     odesolverstate* state,
     ae_state *_state)
{
    ae_int_t i;
    double v;

    _odesolverstate_clear(state);


    /*
     * Підготовка RComm
     */
    ae_vector_set_length(&state->rstate.ia, 5+1, _state);
    ae_vector_set_length(&state->rstate.ba, 0+1, _state);
    ae_vector_set_length(&state->rstate.ra, 5+1, _state);
    state->rstate.stage = -1;
    state->needdy = ae_false;
```

```
    /*
     * перевірка параметрів
     */
    if( (n<=0||m<1)||ae_fp_eq(eps,0) )
    {
        state->repterminationtype = -1;
        return;
    }
    if( ae_fp_less(h,0) )
    {
        h = -h;
    }

    /*
     * швидкий вихід за необхідності
     * після цього блоку ми припускаємо, що  M>1
     */
    if( m==1 )
    {
        state->repnfev = 0;
        state->repterminationtype = 1;
        ae_matrix_set_length(&state->ytbl, 1, n, _state);
        ae_v_move(&state->ytbl.ptr.pp_double[0][0], 1, &y->ptr.p_double[0], 1,
ae_v_len(0,n-1));
        ae_vector_set_length(&state->xg, m, _state);
        ae_v_move(&state->xg.ptr.p_double[0], 1, &x->ptr.p_double[0], 1,
ae_v_len(0,m-1));
        return;
    }

    /*
     * перевірка знову правильний порядок X[]
     */
    if( ae_fp_eq(x->ptr.p_double[1],x->ptr.p_double[0]) )
    {
        state->repterminationtype = -2;
        return;
    }
    for(i=1; i<=m-1; i++)
    {
        if( (ae_fp_greater(x->ptr.p_double[1],x-
>ptr.p_double[0])&&ae_fp_less_eq(x->ptr.p_double[i],x->ptr.p_double[i-
1]))||(ae_fp_less(x->ptr.p_double[1],x->ptr.p_double[0])&&ae_fp_greater_eq(x-
>ptr.p_double[i],x->ptr.p_double[i-1])) )
        {
            state->repterminationtype = -2;
            return;
        }
    }

    /*
     * авто-вибір H за необхідності
     */
    if( ae_fp_eq(h,0) )
    {
        v = ae_fabs(x->ptr.p_double[1]-x->ptr.p_double[0], _state);
        for(i=2; i<=m-1; i++)
        {
            v = ae_minreal(v, ae_fabs(x->ptr.p_double[i]-x->ptr.p_double[i-1],
_state), _state);
        }
        h = 0.001*v;
    }

    /*
     * збереження параметрів
     */
    state->n = n;
```

```
    state->m = m;
    state->h = h;
    state->eps = ae_fabs(eps, _state);
    state->fraceps = ae_fp_less(eps,0);
    ae_vector_set_length(&state->xg, m, _state);
    ae_v_move(&state->xg.ptr.p_double[0], 1, &x->ptr.p_double[0], 1,
ae_v_len(0,m-1));
    if( ae_fp_greater(x->ptr.p_double[1],x->ptr.p_double[0]) )
    {
        state->xscale = 1;
    }
    else
    {
        state->xscale = -1;
        ae_v_muld(&state->xg.ptr.p_double[0], 1, ae_v_len(0,m-1), -1);
    }
    ae_vector_set_length(&state->yc, n, _state);
    ae_v_move(&state->yc.ptr.p_double[0], 1, &y->ptr.p_double[0], 1,
ae_v_len(0,n-1));
    state->solvertype = solvertype;
    state->repterminationtype = 0;

    /*
     * розміщення масивів
     */
    ae_vector_set_length(&state->y, n, _state);
    ae_vector_set_length(&state->dy, n, _state);
}


ae_bool _odesolverstate_init(odesolverstate* p, ae_state *_state, ae_bool
make_automatic)
{
    if( !ae_vector_init(&p->yc, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init(&p->escale, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init(&p->xg, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init(&p->y, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init(&p->dy, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !ae_matrix_init(&p->ytbl, 0, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init(&p->yn, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init(&p->yns, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init(&p->rka, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init(&p->rkc, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init(&p->rkcs, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !ae_matrix_init(&p->rkb, 0, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !ae_matrix_init(&p->rkk, 0, 0, DT_REAL, _state, make_automatic) )
        return ae_false;
    if( !_rcommstate_init(&p->rstate, _state, make_automatic) )
        return ae_false;
    return ae_true;
}


ae_bool _odesolverstate_init_copy(odesolverstate* dst, odesolverstate* src,
ae_state *_state, ae_bool make_automatic)
{
    dst->n = src->n;
```

```
    dst->m = src->m;
    dst->xscale = src->xscale;
    dst->h = src->h;
    dst->eps = src->eps;
    dst->fraceps = src->fraceps;
    if( !ae_vector_init_copy(&dst->yc, &src->yc, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init_copy(&dst->escale, &src->escale, _state, make_automatic)
)
        return ae_false;
    if( !ae_vector_init_copy(&dst->xg, &src->xg, _state, make_automatic) )
        return ae_false;
    dst->solvertype = src->solvertype;
    dst->needdy = src->needdy;
    dst->x = src->x;
    if( !ae_vector_init_copy(&dst->y, &src->y, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init_copy(&dst->dy, &src->dy, _state, make_automatic) )
        return ae_false;
    if( !ae_matrix_init_copy(&dst->ytbl, &src->ytbl, _state, make_automatic) )
        return ae_false;
    dst->repterminationtype = src->repterminationtype;
    dst->repnfev = src->repnfev;
    if( !ae_vector_init_copy(&dst->yn, &src->yn, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init_copy(&dst->yns, &src->yns, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init_copy(&dst->rka, &src->rka, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init_copy(&dst->rkc, &src->rkc, _state, make_automatic) )
        return ae_false;
    if( !ae_vector_init_copy(&dst->rkcs, &src->rkcs, _state, make_automatic) )
        return ae_false;
    if( !ae_matrix_init_copy(&dst->rkb, &src->rkb, _state, make_automatic) )
        return ae_false;
    if( !ae_matrix_init_copy(&dst->rkk, &src->rkk, _state, make_automatic) )
        return ae_false;
    if( !_rcommstate_init_copy(&dst->rstate, &src->rstate, _state,
make_automatic) )
        return ae_false;
    return ae_true;
}


void _odesolverstate_clear(odesolverstate* p)
{
    ae_vector_clear(&p->yc);
    ae_vector_clear(&p->escale);
    ae_vector_clear(&p->xg);
    ae_vector_clear(&p->y);
    ae_vector_clear(&p->dy);
    ae_matrix_clear(&p->ytbl);
    ae_vector_clear(&p->yn);
    ae_vector_clear(&p->yns);
    ae_vector_clear(&p->rka);
    ae_vector_clear(&p->rkc);
    ae_vector_clear(&p->rkcs);
    ae_matrix_clear(&p->rkb);
    ae_matrix_clear(&p->rkk);
    _rcommstate_clear(&p->rstate);
}


ae_bool _odesolverreport_init(odesolverreport* p, ae_state *_state, ae_bool
make_automatic)
{
    return ae_true;
}
```

```
ae_bool _odesolverreport_init_copy(odesolverreport* dst, odesolverreport* src,
ae_state *_state, ae_bool make_automatic)
{
    dst->nfev = src->nfev;
    dst->terminationtype = src->terminationtype;
    return ae_true;
}


void _odesolverreport_clear(odesolverreport* p)
{
}


}
```

// diffequations.h - заголовний файл для модуля обчислення диференціальних рівнянь, використовуваних у модулі розпізнавання голосових команд

```
#ifndef _diffequations_pkg_h
#define _diffequations_pkg_h
#include "ap.h"
#include "alglibinternal.h"

// Декларації обчислювального ядра (типи даних)

namespace alglib_impl
{
typedef struct
{
    ae_int_t n;
    ae_int_t m;
    double xscale;
    double h;
    double eps;
    ae_bool fraceps;
    ae_vector yc;
    ae_vector escale;
    ae_vector xg;
    ae_int_t solvertype;
    ae_bool needdy;
    double x;
    ae_vector y;
    ae_vector dy;
    ae_matrix ytbl;
    ae_int_t repterminationtype;
    ae_int_t repnfev;
    ae_vector yn;
    ae_vector yns;
    ae_vector rka;
    ae_vector rkc;
    ae_vector rkcs;
    ae_matrix rkb;
    ae_matrix rkk;
    rcommstate rstate;
} odesolverstate;
typedef struct
{
    ae_int_t nfev;
    ae_int_t terminationtype;
} odesolverreport;

}

// цей розділ містить C++ інтерфейс
namespace alglib
{

/*************************************************************************

*************************************************************************/
class _odesolverstate_owner
{
public:
    _odesolverstate_owner();
    _odesolverstate_owner(const _odesolverstate_owner &rhs);
    _odesolverstate_owner& operator=(const _odesolverstate_owner &rhs);
    virtual ~_odesolverstate_owner();
    alglib_impl::odesolverstate* c_ptr();
    alglib_impl::odesolverstate* c_ptr() const;
protected:
    alglib_impl::odesolverstate *p_struct;
};
class odesolverstate : public _odesolverstate_owner
{
```

```
public:
    odesolverstate();
    odesolverstate(const odesolverstate &rhs);
    odesolverstate& operator=(const odesolverstate &rhs);
    virtual ~odesolverstate();
    ae_bool &needdy;
    real_1d_array y;
    real_1d_array dy;
    double &x;

};


/*************************************************************************

*************************************************************************/
class _odesolverreport_owner
{
public:
    _odesolverreport_owner();
    _odesolverreport_owner(const _odesolverreport_owner &rhs);
    _odesolverreport_owner& operator=(const _odesolverreport_owner &rhs);
    virtual ~_odesolverreport_owner();
    alglib_impl::odesolverreport* c_ptr();
    alglib_impl::odesolverreport* c_ptr() const;
protected:
    alglib_impl::odesolverreport *p_struct;
};
class odesolverreport : public _odesolverreport_owner
{
public:
    odesolverreport();
    odesolverreport(const odesolverreport &rhs);
    odesolverreport& operator=(const odesolverreport &rhs);
    virtual ~odesolverreport();
    ae_int_t &nfev;
    ae_int_t &terminationtype;

};

/*************************************************************************


Розв'язання диф.рівняння Y'=f(Y,x) із початковими умовами  Y(xs)=Ys
(Y може бути скалярною змінною або вектором N змінних).

ВХІДНІ ПАРАМЕТРИ:
    Y       -   початкові умови, масив[0..N-1].
                містить значення Y[] при X[0]
    N       -   розмір системи
    X       -   точки, в яких Y табулюється, масив[0..M-1]
                інтегрування починається в X[0], закінчується в X[M-1],
                проміжні значення при X[i] повертаються також.
                МАЮТЬ БУТИ ВПОРЯДКОВАНІ ЗА ЗРОСТАННЯМ, АБО ЗА СПАДАННЯМ!!!
    M       -   число проміжних точок + перша + остання точки:
                * M>2 означає, що необхідні як Y(X[M-1]), так і M-2 значень
                  проміжних точок
                * M=2 означає, що слід інтегрувати від X[0] до X[1],
                  Не цікавлячись проміжними значеннями.
                * M=1 означає, що ви не хочете інтегрувати
                  це вироджений випадок, але він обробляється правильно.
                * M<1 означає помилку
    Eps     -   допустиме відхилення (абсолютна/відносна похибка
                на кожному кроці менше Eps.
                * Eps>0, розглядається абсолютна похибка
                * Eps<0, розглядається відносна похибка
```

```
    H       -   початкова довжина кроку, налаштовується автоматично
                після першого кроку. Якщо H=0, крок буде вибрано
                автоматично (звичайно він буде дорівнювати 0.001 від
                min(x[i]-x[j])).

ВИХІДНІ ПАРАМЕТРИ
    State   -   структура, яка зберігає стан алгоритму між послідовними
                викликами OdeSolverIteration. Використовується для зворотнього
                Зв'язку. Ця структура має слідувати за підпрограмою
                OdeSolverIteration.




*************************************************************************/
void odesolverrkck(const real_1d_array &y, const ae_int_t n, const real_1d_array
&x, const ae_int_t m, const double eps, const double h, odesolverstate &state);
void odesolverrkck(const real_1d_array &y, const real_1d_array &x, const double
eps, const double h, odesolverstate &state);


/*************************************************************************
Функція інтерфейсу зворотнього зв'язку
*************************************************************************/
bool odesolveriteration(const odesolverstate &state);


/*************************************************************************
Запуск ітерацій ODE solver

Параметри:
    diff    -   зворотній зв'язок, що обчислює dy/dx для даних y та x
    ptr     -   опціональний показчик, який вказує на diff може бути NULL



*************************************************************************/
void odesolversolve(odesolverstate &state,
    void (*diff)(const real_1d_array &y, double x, real_1d_array &dy, void
*ptr),
    void *ptr = NULL);


/*************************************************************************
Результати роботи ODE solver



ВХІДНІ ПАРАМЕТРИ:
    State   -   стан алгоритму (використовується в OdeSolverIteration).

ВИХІДНІ ПАРАМЕТРИ:
    M       -   кількість табульованих значень, M>=1
    XTbl    -   масив[0..M-1], значення X
    YTbl    -   масив[0..M-1,0..N-1], значення Y в X[i]
    Rep     -   звіт:
                * Rep.TerminationType завершення коду:
                    * -2    X не впорядкований по зростанню/спаданню або
                            є однакові значення X[], тобто X[i]=X[i+1]
                    * -1    задані неправильні параметри
                    *  1    задача розв'язана
                * Rep.NFEV містить кількість обчислень функції

*************************************************************************/
void odesolverresults(const odesolverstate &state, ae_int_t &m, real_1d_array
&xtbl, real_2d_array &ytbl, odesolverreport &rep);
}
```

```
// Цей розділ містить декларації обчислювального ядра (функції)
namespace alglib_impl
{
void odesolverrkck(/* Real    */ ae_vector* y,
     ae_int_t n,
     /* Real    */ ae_vector* x,
     ae_int_t m,
     double eps,
     double h,
     odesolverstate* state,
     ae_state *_state);
ae_bool odesolveriteration(odesolverstate* state, ae_state *_state);
void odesolverresults(odesolverstate* state,
     ae_int_t* m,
     /* Real    */ ae_vector* xtbl,
     /* Real    */ ae_matrix* ytbl,
     odesolverreport* rep,
     ae_state *_state);
ae_bool _odesolverstate_init(odesolverstate* p, ae_state *_state, ae_bool
make_automatic);
ae_bool _odesolverstate_init_copy(odesolverstate* dst, odesolverstate* src,
ae_state *_state, ae_bool make_automatic);
void _odesolverstate_clear(odesolverstate* p);
ae_bool _odesolverreport_init(odesolverreport* p, ae_state *_state, ae_bool
make_automatic);
ae_bool _odesolverreport_init_copy(odesolverreport* dst, odesolverreport* src,
ae_state *_state, ae_bool make_automatic);
void _odesolverreport_clear(odesolverreport* p);

}
#endif
```