Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
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 розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22МЗ
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Пойченко О.П.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Лисенко І.А.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань _12 _____*"Інформаційні технології"*_____
Спеціальність 122 *"Комп'ютерні науки"*
Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

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

Пойченку Олександру Петровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки*

2. Керівник роботи *Лисенко Ірина Анатоліївна, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 37-13 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 розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки*

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

Пойченко О.П. Дослідження та програмна реалізація системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Метою розробки є дослідження та програмна реалізація системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Об'єктом дослідження є процес розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Предметом дослідження є методи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Методи дослідження базуються на методах комп'ютерної графіки, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

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 Delphi 10.4.1.

Ключові слова: комп'ютерні науки, когнітивна графіка

ABSTRACT

Poichenko O.P. Research and software implementation of a system for recognizing graphic images on satellite images using cognitive graphics. 122 Computer Science. 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 system of recognizing graphic images on satellite images using cognitive graphics.

The purpose of the development is research and software implementation of a system for recognizing graphic images on satellite images using cognitive graphics.

The object of research is the process of recognizing graphic images on satellite images using cognitive graphics.

The subject of the study is methods of recognizing graphic images on satellite images using cognitive graphics.

Research methods are based on computer graphics methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the system for recognizing graphic images on satellite images using cognitive graphics.

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 Delphi 10.4.1 environment.

Keywords: computer science, cognitive graphics

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
ПЗ	–	програмне забезпечення

ВСТУП

Актуальність теми. Об'єктом додатка методів розпізнавання графічних образів і когнітивної графіки з'явилася перспективна система, орієнтована на обробку інформації із супутників. Перелічимо деякі завдання обробки космічної інформації:

– виявлення локальних об'єктів на аерокосмічних знімках;

– кластеризація й розпізнавання цільових об'єктів;

– визначення місця розташування об'єкта в заданій системі координат;

– стиск і відновлення графічної інформації;

– фільтрація;

– прогнозування даних телеметрії (тимчасових рядів);

– виявлення несправностей і позаштатних ситуацій.

Технологію первинної обробки інформації становлять хвильові алгоритми виділення об'єктів на знімках, методи видалення свідомо помилкових об'єктів і нормалізації претендентів на розпізнавання. Велике значення для якості роботи нейронних мереж має приведення графічних об'єктів до стандартного виду в змісті орієнтації й масштабу. Нейронні мережі використовуються в самому кінці технологічного ланцюжка, причому від якості передобробки й типу нейронної мережі значною мірою залежить результат розпізнавання. Це пов'язане з великою чутливістю нейронні мережі до наявності шумів, положенню й масштабу образів і т.д. Крім типових мереж можна формувати й спеціальні мережі. Результати роботи нейронних мереж (в основному використовувалися мережі прямого поширення, Хеммінга й Кохонена): приблизно 60%-80% правильного розпізнавання. Результат вдається трохи поліпшити за рахунок, застосування комітетів нейронні мережі.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

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

– Огляд існуючих систем розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

– Дослідження системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

– Програмна реалізація системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Об'єктом дослідження є процес розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Предметом дослідження є методи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Методи дослідження базуються на методах комп'ютерної графіки, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

– Розроблено вітчизняний продукт розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

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 розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки, 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

Система призначена для розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки. Для поліпшення результатів відділення цільових об'єктів від помилкових використаний комплекс методів спеціальної обробки, у тому числі методи виділення контурів, стиску простору ознак, виділення «кістякового зображення» і ін.

Так, наприклад, завдання визначення повітряної мети зажадало використання технології виділення контурів, обчислення інваріантних моментів і застосування узагальненої метрики Евкліда-Махаланобіса.

Важлива по значимості прикладне завдання – виділення регіонів. Регіон – це область на космічному знімку, що представляє з ряду причин інтерес для користувача. Запропонована технологія формування еталонних текстур і узагальнена метрика вирішують досить упевнено поставлене завдання навіть без знання спектральних характеристик точок поверхні, одержуваних із супутників у результаті дистанційного зондування Землі.

Узагальнена метрика є універсальною. Вона на відміну від метрики Махаланобіса застосовна у випадках, коли виділювана область містить зовсім однакові або дуже близькі по яскравості пікселі, тобто коли немає розкиду яскравісних параметрів.

Інша не менш важливе завдання – стиск і фільтрація графічної інформації. Фільтрація здійснюється мережею Хопфілда, а стиск мережею Кохонена. Мережа Кохонена програє за інших рівних умов алгоритмуJPEG-2000, однак тут має місце елемент захисту інформації, так як без знання настроювань мережі розшифрувати цільову інформацію неможливо.

1.2 Область застосування

Зараз тема супутникових знімків часто піднімається в розмовах, далеких від космосу. При цьому з обговорення до обговорення перемиваються древні міфи Холодної війни, про "прочитати номер машини" і "порахувати зірочки на погонах" або навіть "прочитати газету".

Сьогодні ми на конкретних прикладах розглянемо на що здатна космічна оптика, і чи все видно зверху.

Для початку, невелике відкриття для багатьох: в Google Map немає супутникових знімків дозволом вище 50 см на піксель. Донедавна, комерційне поширення більше детальних супутникових знімків було заборонено в США. Тому якщо ви знайшли в якому-небудь місті знімки, де видні люди, що гуляють, і інші подробиці – це аерофотознімки, його публікувати можна.

Таке протиріччя довго не влаштовувало космічних приватників, і вони все-таки пролоббіювали послаблення закону, і тепер можна продавати знімки дозволом до 25 см на піксель. На сьогодні це межа комерційної супутникової зйомки.

Але навіть для таких знімків потрібна складна техніка. От, наприклад, супутник WorldView-3 компанії DigitalGlobe: дозвіл 31 см, діаметр дзеркала телескопа 1,1 м, вартість $650 млн доларів.

Не дуже давно компанія вперше опублікувала пробні знімки дозволом 40 см. На сьогодні це самі детальні супутникові знімки, які коли-або були легально й відкрито опубліковані.

Для приклада DigitalGlobe виклали знімки Мадрида. Як видно, можна розглянути безліч подробиць: легкові машини просто відрізнити від вантажних, навіть, здається, що купаються людей у басейнах можна розглянути у вигляді точок. Але Мадрид обраний невипадково: чим ближче до екватора, тим менше хмарність. Ще для демонстрації можливостей супутників часто вибирають Дубай – там багато всяких колоритних об'єктів, і пустельна погода сприяє

спостереженню. Колосальні витрати на створення приватних супутників, здатних на таку якість зйомки, викликають закономірне питання: як вони окупаються. У цьому секрету немає: більше 50% замовлень компанії DigitalGlobe ідуть із Пентагона. Інше від Google і індивідуальних замовників.

Але це однаково комерційні супутники, а що ж можуть військові.

Отут усе складніше, але в цілому цілком передбачувано. Легендарний і самий потужний американський супутник-шпигун ставиться до серії Keyhole-11. Вірогідно про нього мало що відомо, навіть вигляд не до кінця прояснений, хоча астрономи-аматори періодично "перехоплюють" його.

Зате відомо, що космічний телескоп Hubble створювався на виробничій лінії, з якої раніше сходили супутники-шпигуни, а пару років тому американський шпигунський відділ (National Reconnaissance Office) подарував NASA два телескопи діаметром 2,4 метри, які завалялися на складі.

Тому, найбільше імовірно, КН-11 має дзеркало діаметром 2,4 метри, як і відомий космічний телескоп Hubble.

Шляхом нескладного порівняння з WorldView-3, у якого дзеркало 1,1 метр, ми одержуємо, що якість шпигунських знімків повинне бути приблизно в 2,3 рази краще. Але є різниця: WorldView-3 літає на висоті 617 км, а наймолодший КН-11 (за назвою USA-245) на висоті від 270 до 970 км.

Космічний телескоп Hubble з висоти 700 км міг би зняти Землю з дозволом до 10-15 см, в ідеальних умовах, якби йому дозволяли технічні можливості. Відповідно, КН-11 у нижній точці своєї орбіти здатний дати дозвіл до 5 см. Але, знов-таки – це в ідеальних умовах, під час відсутності хмарності, смогу, туману й просто пилу над об'єктом зйомки. Крім того, чим вище дозвіл, і чим ближче супутник до поверхні Землі, тим уже смуга захвата його зйомки й менше можливості подивитися по сторонах. Тобто таку зйомку доцільно застосовувати тільки по заздалегідь розвіданих об'єктах, у ясну погоду, і тільки під час, що обумовлено орбітою апарата.

Тому американські військові й платять американським комерсантам, що своїх технічних засобів не вистачає, і простіше купити потрібні знімки, чим створювати купу супутників, кожний вартістю з авіаносець.

Таким чином, в ідеальних умовах, теоретично, усього один супутник-шпигун здатний розглянути планку номерного знака на машині у вигляді декількох білих пікселів. Але прочитати номер, не говорячи вже про погони й газети – неможливо просто фізично.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки, 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

ABBYY Finereader

Про FineReader чули, напевно, більшість із вас. Ця програма краща або одна із кращих для якісного розпізнавання текстів українською мовою. Програма є платною й ціна ліцензії для домашнього використання становить ледве менш 700 грн. Також є можливість скачати пробну версію FineReader або ж скористатися онлайн розпізнаванням текстів в ABBYY Fine Reader Online (безкоштовно можна розпізнати кілька сторінок, далі – платно).

Установка пробної версії FineReader не викликала ніяких проблем. ПЗ може інтегруватися з Microsoft Office і Провідником Windows, для того щоб було зручніше запустити розпізнавання. З обмежень безкоштовної пробної версії – 15 діб використання й можливість розпізнати не більше 50 сторінок.

Так як сканера в мене немає, то для перевірки я скористався знімком з неякісної камери телефону, у якому небагато відредагував контрастність. Якість ні до чого не придатна, подивимося, як впорається.

FineReader може одержувати графічне зображення тексту прямо зі сканера, із графічних файлів або камери. У моєму випадку, досить було відкрити файл зображення. Результат порадував – усього пари помилок. Відразу скажу, що це кращий результат із всіх перевірених програм при роботі з даним зразком – схожа якість розпізнавання бути тільки на безкоштовному онлайн сервісі Free Online OCR (але в цьому огляді ми говоримо тільки про програмні засоби, не онлайн розпізнаванні).

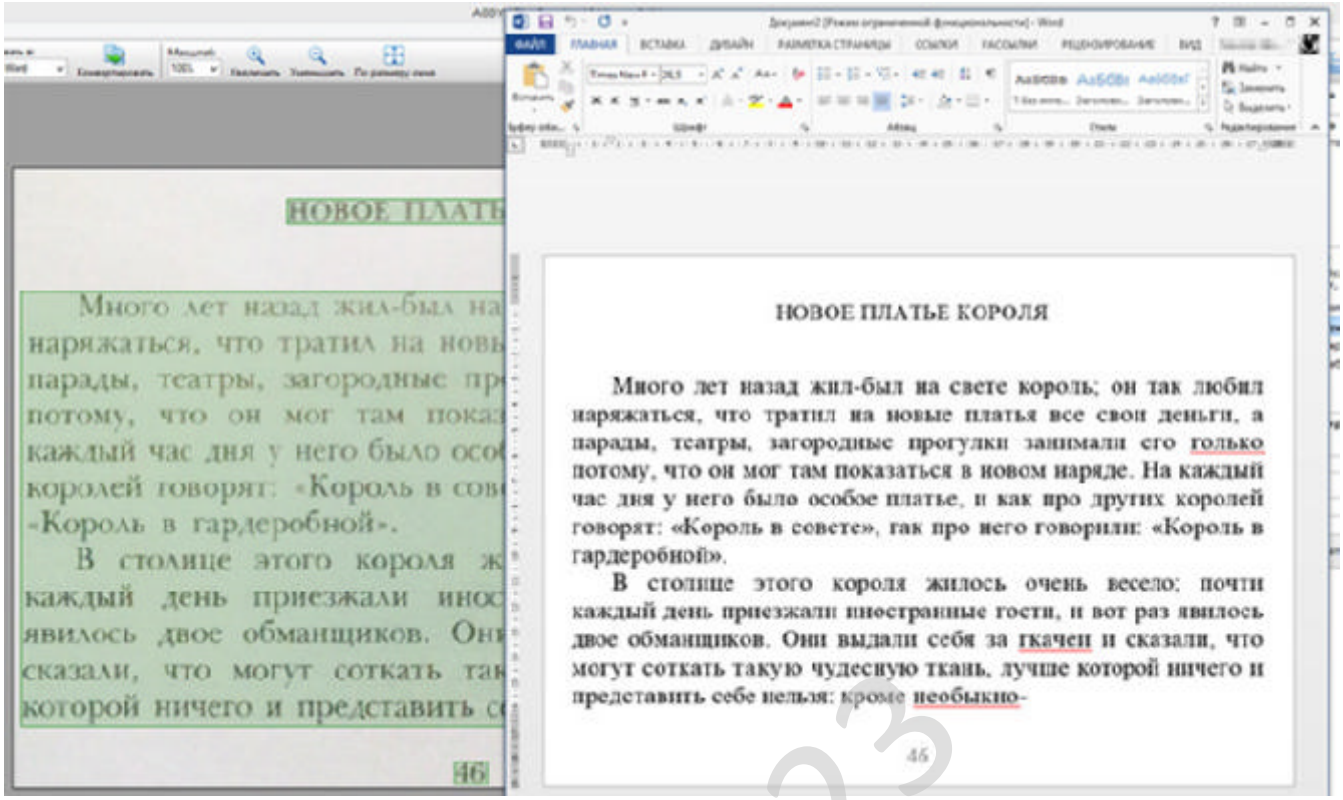

Рисунок 2.1 – Результат розпізнавання тексту в FineReader

Відверто говорячи, для кириличних текстів в FineReader, напевно, немає конкурентів. Плюсами програми є не тільки якість розпізнавання текстів, але й широка функціональність, підтримка форматування, грамотний експорт у безліч форматів, включаючи Word docx, pdf і інші можливості. Таким чином, якщо завдання OCR – це те, із чим зіштовхуєтеся постійно, то не пошкодуйте порівняно невеликої кількості грошей і це цілком окупиться: ви заощадите величезну кількість часу, швидко одержуючи якісний результат в FineReader. Я, до речі, не рекламую нічого – дійсно вважаю, що тим, кому потрібно розпізнати більше десятка сторінок, варто задуматися про покупку такого ПЗ.

CuneiForm

По моїй оцінці, друга по популярності програма OCR в Україні – безкоштовна CuneiForm, скачати яку можна з офіційного сайту http://cognitiveforms.ru/products/cuneiform/.

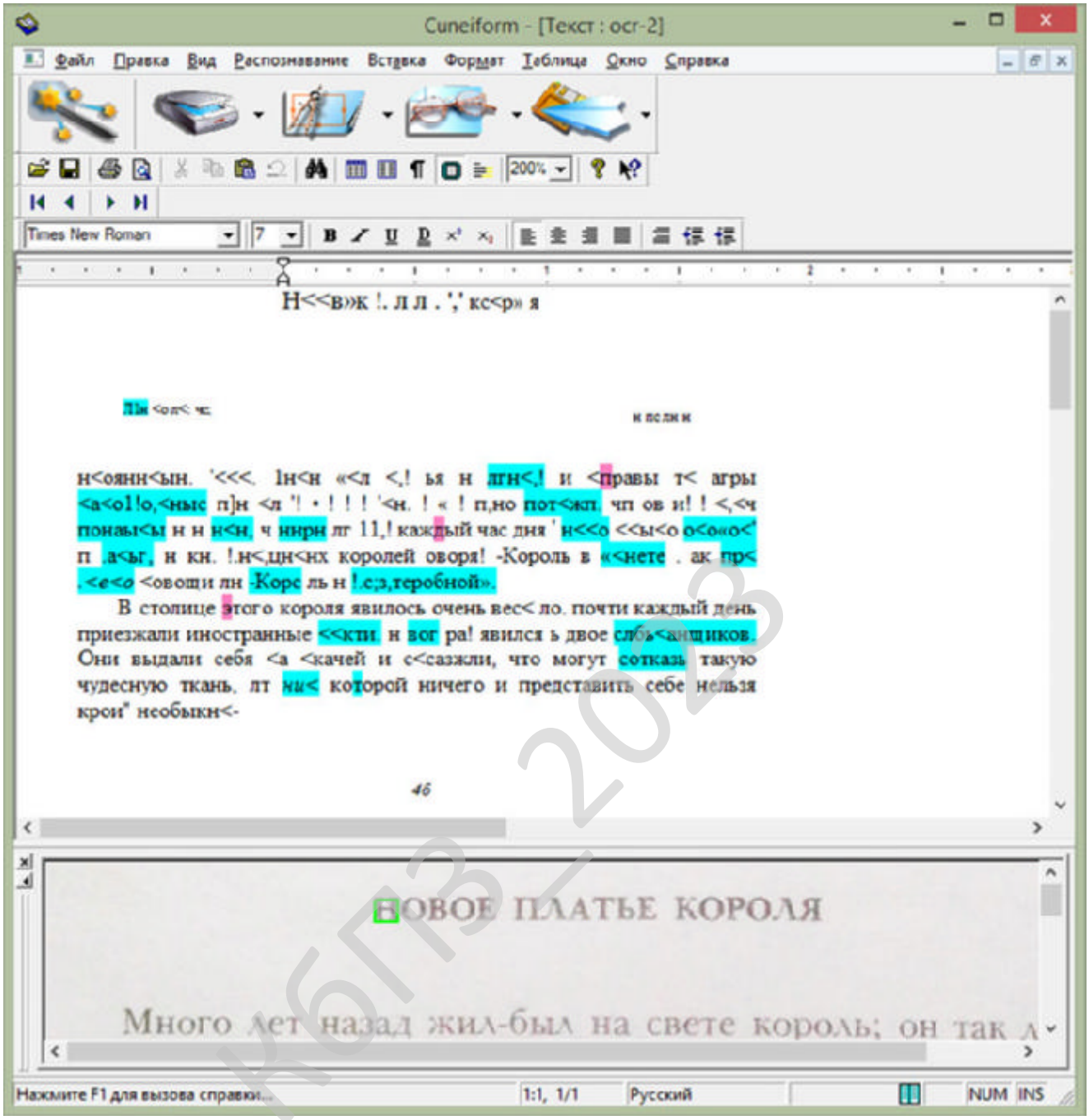

Рисунок 2.2 – Розпізнавання тексту в CuneiForm

Установка програми також дуже проста, ніякого стороннього софта вона встановити не намагається. Інтерфейс лаконічний і зрозумілий. У деяких випадках найпростіше скористатися майстром, для чого призначена перша з іконок у меню.

Зі зразком, яким я користувався в FineReader, програма не впоралася, або, точніше, видала щось читаєме погано й ошмітки слів. Друга спроба була почата зі скріншотом тексту із сайту самої цієї програми, що, щоправда, довелося збільшити (їй потрібні скани з дозволом 200dpi і вище, скриншоти з товщиною ліній шрифтів 1-2 пікселя вона не читає). Отут вона впоралася добре (частина тексту не розпізнана, так як була обрана тільки українська мова).

Таким чином, можна припустити, що CuneiForm – це те, що варто спробувати, особливо якщо у вас якісно відскановані сторінки й ви хочете розпізнати їх безкоштовно.

Microsoft OneNote

До складу Microsoft Office, починаючи з версії 2007 і закінчуючи поточною, 2013, є присутнім програма для ведення заміток – OneNote. У ній також присутні функції розпізнавання тексту. Для того, щоб скористатися нею, просто вставте відскановане або будь-яке інше зображення тексту в замітку, кликніть правою клавішею миші по ній і скористайтеся контекстним меню. Відзначу, що за замовчуванням для розпізнавання встановлена англійська мова.

Не можу сказати, що текст розпізнається ідеально, але, наскільки я можу судити, трохи краще навіть ніж в CuneiForm. Плюс програми, як уже було сказано, у тому, що із чималою ймовірністю вона вже встановлена на вашім комп'ютері. Хоча, звичайно, її використання якщо буде потреба роботи з більшою кількістю відсканованих документів чи навряд буде зручним, скоріше вона підійде для швидкого розпізнавання візиток.

OmniPage Ultimate, OmniPage 18

Я не знаю, наскільки гарна програма для розпізнавання текстів OmniPage: пробних версій немає, десь завантажувати не хочу. Але, якщо її ціна виправдана, а вона обійдеться приблизно в 2000 грн. у версії для індивідуального використання й не Ultimate, те це повинне бути щось вражаюче. Сторінка програми: http://www.nuance.com/ for-individuals/by-product/omnipage/index.htm.

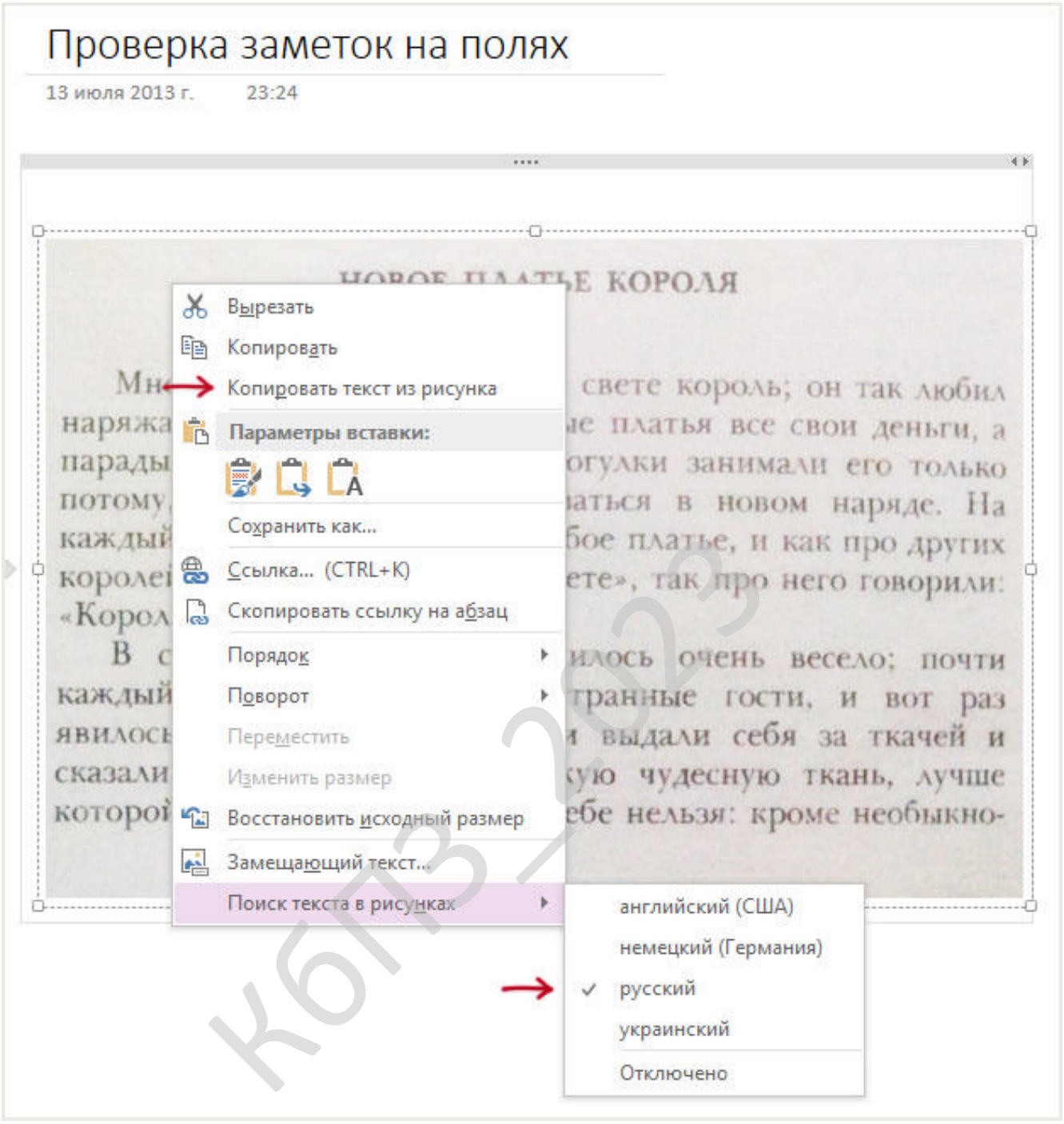

Рисунок 2.3 – Розпізнавання в Microsoft OneNote

Якщо ознайомитися з характеристиками й відкликаннями, у тому числі й в українськомовних виданнях, у них відзначається, що OmniPage дійсно забезпечує якісне й точне розпізнавання, у тому числі й українською мовою, порівняно легко розбирає не самі якісні скани й надає набір додаткових інструментів. З недоліків виділяють не самий зручний, особливо для починаючого користувача, інтерфейс.

Так чи інакше, на західному ринку OmniPage – прямій конкурент FineReader і в англомовних рейтингах вони борються саме між собою, а тому, думаю, програма повинна бути гідною.

Це далеко не всі програми даного типу, існують також різні варіанти невеликих безкоштовних програм, але, поки експериментував з ними знайшов два головних недоліки їм властивих: відсутність підтримки кирилиці, або різне, не занадто корисне ПЗ у комплекті установки, а тому вирішив не згадувати їх тут.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium,

використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4

забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні

64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

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 розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

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

Когнітивна графіка

У цей час немає єдиних принципів когнітивного відображення інформації, але є розуміння того факту, що графічні образи здатні нести в собі в стислій і одночасно із цим доступну для користувача формі інформацію достатню для ухвалення адекватного рішення. Кожний образ створюється індивідуально з обліком конкретної прикладної області, вивчається в процесі життєвого циклу об'єкта й інтерпретується експертом з використанням накопичених знань. Багатомірні дані за допомогою ЕОМ можуть бути співвіднесені в когнітивний графічний образ у вигляді інтегральних функціональних профілів або сцен, що відбивають особливості стану об'єкта. Єдиний математичний апарат аналізу й загальні методи візуалізації багатомірних даних у цей час відсутні. Очевидно, мова може йти про інтеграцію й оптимізацію таких подань стосовно до конкретних прикладних областей.

Для побудови схеми рішення завдання розпізнавання образів зручно користуватися засобами графічного інтерфейсу, які дозволяють не тільки формувати алгоритм обробки даних підключенням відповідних виконавчих модулів, але й відслідковувати порядок рішення в динаміку шляхом колірного підсвічування відповідних зв'язків.

Для контролю настроювання нейронні мережі з невеликим числом нейронів застосовується спеціальний графічний динамічний образ. Таке подання дозволяє бачити стан мережі, знаки коефіцієнтів (синій і червоний кольори) і величини вагових коефіцієнтів, шляхом їхнього відображення відтінками синього й червоного кольору.

Космічна зйомка

Носії й космічні комплекси

Космозйомка займає одне із провідних місць серед різних методів дистанційного зондування. Вона здійснюється за допомогою:

– штучні супутники Землі (ШСЗ);

– міжпланетні автоматичні станції;

– довгострокові орбітальні станції;

– пілотовані космічні кораблі.

Космічні системи (комплекси) моніторингу навколишнього середовища містять у собі (і виконують):

1. Супутникові системи на орбіті (центр керування польотами й зйомкою).

2. Прийом інформації наземними пунктами прийому, супутниками-ретрансляторами.

3. Зберігання й поширення матеріалів (центри первинної обробки, архіви знімків). Розроблено інформаційну пошукову систему, що забезпечує нагромадження й систематизацію матеріалів, одержуваних зі штучних супутників Землі.

Орбіти космічних літальних апаратів

Орбіти носіїв діляться на 3 типи:

– екваторіальні;

– полярні (полюсні);

– похилі.

Орбіти підрозділяють на:

– кругові (точніше, близькі до кругового). Космознімки, отримані з космічного носія, що рухався по круговій орбіті, мають приблизно однаковий масштаб.

– еліптичні.

Орбіти розрізняють також по положенню відносно Землі або Сонця:

– геосинхроні (відносно Землі);

– геліосинхроні (щодо Сонця).

Геосинхроні – космічний літальний апарат рухається з кутовою швидкістю, рівної швидкості обертання Землі. Це створює ефект “зависання” космічного носія в одній точці, що зручно для постійних зйомок того самого ділянки земної поверхні.

Геліосинхроні (або сонячно-синхронні) – космічний апарат проходить над певними ділянками земної поверхні в той саме місцевий час, що використовується при виробництві багаторазових зйомок при однакових умовах висвітлення. Геліосинхроні орбіти – орбіти, при зйомці з яких сонячна освітленість земної поверхні (висота Сонця) залишається практично незмінної досить тривалий час (майже протягом Сезону). Це досягається наступним шляхом. Оскільки площина будь-якої орбіти під впливом несферичності Землі злегка розвертається (прецесує), то виявляється можливим, підбираючи певне співвідношення нахилення й висоти орбіти, домогтися, щоб величина прецесії була рівною добовому повороту Землі навколо Сонця, тобто близько 1° у добу. Серед навколоземних орбіт вдається створити лише трохи сонячно-синхронних, нахилення яких завжди зворотне. Наприклад, при висоті орбіти 1000 км нахилення повинне бути 99°.

Види зйомок

Космічну зйомку ведуть різними методами.

По характері покриття земної поверхні космічними знімками можна виділити наступні зйомки:

– одиночне фотографування;

– маршрутну;

– прицільну;

– глобальну зйомку.

Одиночне (вибіркове) фотографування виконується космонавтами ручними камерами. Знімки звичайно виходять перспективними зі значними кутами нахилу.

Маршрутна зйомка земної поверхні виробляється уздовж траси польоту супутника. Ширина смуги зйомки залежить від висоти польоту й кута огляду знімальної системи.

Прицільна (вибіркова) зйомка призначена для одержання знімків спеціально заданих ділянок земної поверхні осторонь від траси.

Глобальну зйомку роблять із геостаціонарних і полярно-орбітальних супутників. Чотири-п'ять геостаціонарних супутників на екваторіальній орбіті забезпечують практично безперервне одержання дрібномасштабних оглядових знімків всієї Землі (космічне патрулювання) за винятком полярних шапок.

Аерокосмічний знімок

Аерокосмічний знімок – це двовимірне зображення реальних об'єктів, що отримано по певних геометричних і радіометричним (фотометричним) законах шляхом дистанційної реєстрації яскравості об'єктів і призначено для дослідження видимих і схованих об'єктів, явищ і процесів навколишнього світу, а також для визначення їхнього просторового положення.

Космічний знімок по своїх геометричних властивостях принципово не відрізняється від аерофотознімка, але має особливості, пов'язані з:

– фотографуванням з більших висот;

– великою швидкістю руху.

Так як супутник у порівнянні з літаком рухається значно швидше, те вимагає коротких витримок при зйомці.

Космічна зйомка розрізняється по:

– масштабам;

– просторовому дозволу;

– оглядовості;

– спектральним характеристикам.

Ці параметри визначають можливості дешифрування на космічних знімках різних об'єктів і рішення тих геологічних завдань, які доцільно вирішувати з їхньою допомогою.

Типи знімків підрозділяються по оглядовості, по масштабі, по просторовому дозволі.

Масштаб і оглядовість (форма, розмір) космічних знімків дозволяють виявити об'єкти різного рангу, зняті в один час і в одному режимі зйомки.

Оглядовість знімка залежить від розмірів ділянок земної поверхні, відображеної на космознімку, і виміряється в одиницях площі.

Найпоширеніший розмір кадру космічного знімка 18х18 см дозволяє бачити все зображення одночасно, не “переводячи погляду”, без послідовного огляду.

При збільшенні масштабу знімка проекційним шляхом оглядовість знімка зберігається, а рівень генералізації знижується.

По оглядовості (охвату території одним знімком) знімки розділяють:

1. Глобальні, що охоплюють всю планету. Ширина зони охвату більше 10 тис. км, а територіальний охват становить сотні мільйонів квадратних кілометрів.

2. Крупнорегіональні, що відображають материки, їхні частини й великі регіони, – знімки з метеорологічних супутників на навколоземних орбітах, а також знімки малого й середнього дозволу з ресурсних супутників. Ширина зони охвату варіює від 3 тис. км у знімків малого дозволу до 500 км у знімків середнього дозволу, територіальний охват становить мільйони квадратних кілометрів. На одному знімку цього типу зобразиться Західна Європа, майже вся Австралія, Середня Азія, Тибет.

3. Регіональні, на яких зображуються регіони і їхня частини, – це знімки з ресурсних і картографічних супутників, а також з пілотованих кораблів і орбітальних станцій. Найбільш характерний охват 350 х 350 км^2, 180 х 180 км^2, 60 х 60 км^2. На знімку подібного охвату зобразиться така держава, як Бельгія, невелика область, наприклад Київська, великі мегаполіси.

4. Локальні, на яких зображуються відносно невеликі ділянки місцевості, – знімки із супутників для детального спостереження й великомасштабного топографічного картографування з охватом порядку 10 х 10 км^2. На такому

знімку зобразиться промисловий комплекс, велике господарство, невелике місто, а для Києва буде потрібно кілька знімків.

Масштаби космознімків різні: від 1:1000 до 100 000 000, тобто він може мінятися в сто тисяч разів. Найпоширеніші масштаби космічних знімків: від 1:200 000 до 1:10 000 000.

Масштаби космознімків залежать від:

– висоти фотографування;

– фокусної відстані апарата;

– коефіцієнта збільшення;

– кутів нахилу;

– кривизни земної поверхні.

Просторовий дозвіл (або дозвіл на місцевості) визначається розміром найменшого об'єкта (Δ), відтвореного на знімку, і визначається по формулі:

$$\Delta = m/2N$$

де:

m – масштаб знімка;

N – розв'язна здатність знімка, тобто число роздільно фотографічно відтворених чорно-білих штрихів на відрізку довжиною 1 мм.

Розпізнавання об'єктів на знімках залежить від масштабу зйомки й розв'язної здатності. По співвідношенню масштабного ряду космічних знімків з масштабним рядом геологічних карт, прийнятих в Україні, космічні знімки розділяються по рівнях природної генералізації на:

– глобальні;

– континентальні;

– регіональні;

– локальні;

– детальні.

Для скануючих систем дозволу по маршруті й уздовж рядка (краю знімка) відрізняються й можуть змінюватися в кілька разів залежності від кута

сканування, тому при дешифруванні використовують тільки центральну (робочу) смугу. У деяких випадках при збільшенні знімків до мінімального дозволу (до 5 ліній на 1 мм) вони можуть охоплювати кілька рівнів розпізнавання (генералізації).

Знімки глобального рівня

Космічні знімки глобального рівня генералізації одержують із висот 20 – 30 тис. км.:

– з міжпланетних автоматичних станцій;

– високоорбітальних ШСЗ (“Блискавка” і ін.)

Масштаб ряду карт: 1:5 000 000.

Космічні знімки глобального рівня генералізації охоплюють всю або більшу частину півкулі. Вони дозволяють:

– виявляти найбільш протяжні глибинні розлами й зони розламів;

– гігантські кільцеві структури;

– з'ясовувати характер зчленування великих структурних елементів земної кори;

– зв'язок поверхневої геології із глибинною будовою літосфери.

Розробки НАСА космічної системи глобального моніторингу ЕО призначені для комплексного планетарного дистанційного вивчення Землі як єдиної системи (хімічний склад атмосфери, рух хвиль цунамі в океані й т.д.).

Передбачається робота декількох ШСЗ, що передають інформацію кожні 10 хв. у реальному масштабі часу.

Знімки континентального рівня

Космічні знімки континентального рівня генералізації мають малий дозвіл. Їх одержують телевізійними скануючими системами з ШСЗ “Метеор” і ін. Космічні знімки цього рівня генералізації дозволяють:

– установлювати структурно-геологічні особливості великих областей земної кулі;

– виділяти матеріально-структурні комплекси гірських порід;

– глибинні розлами;

– проводити або уточнювати тектонічне районування.

Масштаб ряду карт: 1:5 000 000.

Знімки регіонального рівня

Космічні знімки регіонального рівня генералізації відрізняються середнім дозволом. Їх одержують фотографічними й скануючими системами з ресурсних ШСЗ “Метеор”, “Лендсат”, а також з пілотованих космічних кораблів і довгострокових орбітальних станцій.

Масштаб ряду карт: 1:1 000 000 і 1:500 000.

Знімки локального рівня

Космічні знімки локального рівня генералізації одержують фотографічними системами з пілотованих космічних кораблів довгострокових орбітальних станцій за допомогою високоякісної апаратури типу МКФ-6 і з ресурсного ШСЗ “Лендсат”.

Знімки локального рівня генералізації дозволяють:

– істотно уточнити геологічну структуру різних регіонів;

– представляють основний матеріал для геологічного картування в масштабах 1:500 000 і :1 000 000;

– для складання спеціалізованих тематичних карт геологічного змісту, у тому числі прогнозно-мінералогічних.

Ця зйомка використовує цифрові сканери, що дають високе тривимірне зображення. Одержувані знімки придатні для кадастру й інвентаризації, для виготовлення середньомасштабних і великомасштабних карт.

Масштаб ряду карт: 1:200 000 і 1:100 000.

Знімки детального рівня

Космічні знімки детального рівня генералізації масштабу 1:100 000 і крупніше по своїх властивостях близькі до висотних аерофотознімкам і знімкам дрібного масштабу. Одержують знімки при фотографуванні високоякісними довгофокусними знімальними камерами з орбіт висотою близько 200 км.

Космічні знімки детального рівня генералізації використовують (як і аерофотоматериали) при космофотогеологічних дослідженнях.

Масштаб ряду карт: 1:50 000 і 1:25 000.

Всі аерокосмічні знімки діляться на:

– аналогові (звичайно фотографічні);

– цифрові (електронні).

Цифрові знімки

Зображення цифрових знімків утворено з окремих однакових елементів – пікселів. Яскравість кожного пікселя характеризується одним числом. Аерокосмічний знімок складається з мільйонів пікселів.

Технологічні способи одержання знімків різні.

Комплекс обробки інформації детального дозволу

За матеріалами космічної зйомки детального дозволу розробляються:

1. Цифрові культурзображення районів земної поверхні з лінійним дозволом 1м і 2 м.

2. Цифрові моделі рельєфу

3. Ортотрансформовані цифрові зображення (фотоплани).

4. Топографічні карти різних масштабів.

5. Спеціальні й тематичні карти різних масштабів.

6. Векторні геоінформаційні шари (використовуються при створенні спеціальних геоінформаційних систем).

7. Тематично оброблені матеріали (оцінка екологічної обстановки, обстановки в районах екологічних і техногенних катастроф, урожайності сільськогосподарських , стану лісів і т.д.)

Комплекс тематичної обробки

Тематична обробка космічної інформації дозволяє оцінювати стан навколишнього середовища й природних об'єктів. Багатофункціональний комплекс тематичної обробки дозволяє проводити комплексну обробку даних від різних систем.

Особливості зображення на космічних знімках залежать від впливу факторів: технічних і природних (природних).

Технічні фактори, що впливають на інформативність космічних знімків, є:

а) параметри польоту (траєкторія, висота, тип орбіти, швидкість руху);

б) характеристики космічних знімальних систем (фокусна відстань апарата, спектральний діапазон, що дозволяє здатність знімальних систем;

в) способи обробки матеріалів.

Природні (природні) фактори – електромагнітний спектр Сонця, стан атмосфери, сезон зйомки, ландшафтно-кліматичні особливості території зйомки.

Переваги космозйомки

Супутник, що летить, не випробовує вібрацій і різких коливань, тому космічні знімки вдається одержувати з більше високою розв'язною здатністю й високою якістю зображення, чим аерознімки. Знімки можуть бути переведені в цифрову форму для наступної комп'ютерної обробки.

Недоліки космозйомки

Інформація не піддається автоматизованій обробці без попередніх перетворень. При космофотозйомці відбувається зсув точок (під впливом кривизни Землі), їхня величина на краях знімка досягає 1,5 мм. У межах знімка порушена сталість масштабу, розходження якого на краях і в центрі знімка може становити вище 3%.

Недоліком фотозйомки є його неоперативність, так як контейнер із плівкою спускається на Землю не частіше, ніж один раз у кілька тижнів. Тому фотографічні космічні знімки рідко використовуються для оперативних цілей, а представляють інформацію довгострокового використання.

3.2 Розробка структурної схеми

Програмне забезпечення розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки, побудовано на базі нейронної мережі Хеммінга.

Алгоритм нейронної мережі Хеммінга

Опис реалізації алгоритму.

1. Реалізація мережі. Вхідні символи.

Для визначення загальних параметрів НМ – кілька констант:

– Розмір сторони вихідного зображення.

– Розмір сторони зображення для розпізнавання.

– Скільки входів.

– Скільки образів.

– Вага синапсів другого шару.

Розмір вхідного образа (bmp-файл) повинен бути 100x100 точок, а перед обробкою він приводиться до розміру 40x40. Таким чином, кількість входів мережі – 1600 (один вхід – одна точка зображення; параметр N). Кількість виходів M збігається з кількістю цифр і рівняється 10. Вага негативного зворотного зв'язка другого шару був прийнятий рівним – 0.05.

Кожний нейрон представляється у вигляді простої структури. Шар – це просто масив нейронів. Є два шари.

Тут уже нема рації зберігати значення окремих ваг, так як вони не міняються протягом всієї роботи мережі.

2. Навчання мережі.

Після процедури навчання списки вагових коефіцієнтів синапсів першого шару будуть містити образи еталонних символів.

Алгоритм навчання мережі Хеммінга, адаптований для даного завдання, виглядає так:

2.1. Вибирається i-й вхідний образ.

2.2. Зображення локалізується й приводиться до потрібного масштабу.

2.3. Образ поточечно подається на входи i-го нейрона. Якщо k-а точка образа чорна, то ваги k-го входу привласнюється значення 0.5, у противному випадку – 0.5.

2.4. Перехід на Крок 2.1, поки не вичерпані всі еталонні образи.

3. Локалізація й масштабування зображення.

Для успішної роботи потрібно з'ясувати точний розмір і місце розташування образа. Після локалізації – приводимо образ до розміру 40*40 (масштабування). Ці операції виконуються як на етапі навчання мережі, так і на етапі розпізнавання.

Для еталона, через відсутність перешкод, локалізацію провести дуже просто: досить послідовно переглянути всі точки образа й знайти границі образа:

У даній реалізації використовується стандартний алгоритм зворотного масштабування без інтерполяції.

4. Перекручування зображення.

Щоб не розпізнавати еталонні образи – уведемо перекручування. Додавання N-процентного шуму.

5. Алгоритм розпізнавання.

Загальний алгоритм розпізнавання для мережі Хеммінга складається із чотирьох частин:

5.1. Подача розпізнаваного образа на входи мережі.

5.2. Передача даних з першого шару на другий.

5.3. Обробка даних другим шаром.

5.4. Вибір розпізнаного образа.

5.1. Алгоритм роботи першого етапу виглядає так:

5.1.1. Вибирається черговий нейрон.

5.1.2. Обнуляється його вихід.

5.1.3. Локалізується й приводиться зображення до потрібного масштабу.

5.1.4. Локалізований образ поточечно подається на входи i-го нейрона. Якщо k-я точка образа чорна, то до значення виходу додається значення ваги k-го входу, у противному випадку це значення віднімається.

5.1.5. Значення виходу пропускається через функцію лінійного порога.

5.1.6. Перехід на Крок 5.2, поки не вичерпані всі нейрони першого шару.

.2. Тепер треба передати дані з виходів першого шару на входи другого й у список результатів попереднього проходу розпізнавання.

5.3. Тепер може починати роботу другий шар. Розглянемо алгоритм його роботи:

5.3.1. Обнуляється лічильник ітерацій.

5.3.2. Запам'ятовуються виходи нейронів у списку результатів попереднього проходу.

5.3.3. Перебирається мережа по нейронах.

5.3.4. і-й нейрон посилає свій вихід на і-й вхід кожного нейрона.

5.3.5. Кожний нейрон, приймаючи значення, накопичує їх, попередньо помноживши на коефіцієнт е (крім випадку, коли нейрон приймає своє ж значення – тоді він повинен помножити його на 1).

5.3.6. Перехід на Крок 5.3.4, поки не будуть оброблені всі нейрони.

5.3.7. Накопичені суми кожний нейрон посилає на свій вихід.

5.3.8. Перехід на Крок 5.3.2, поки виходи нейронів на поточній ітерації не збіжаться з виходами на попередній або поки лічильник числа ітерацій не перевищить деяке значення.

Теоретично, другий шар повинен працювати поки його виходи не стабілізуються, але на практиці кількість ітерацій штучно обмежують.

5.4. Останній Крок – вибір з нейронів другого шару з найбільшим значенням на виході. Його номер і є номер розпізнаного образа.

Структурна схема системи наведена на рисунку 3.1. Структурно система складається з наступних частин:

1. База даних журналювання розпізнаних образів. У цю баз даних заносяться усі дані, які відносяться до розпізнаних об’єктів.

2. База даних образів на знімках із супутників. У цій базі даних зберігаються фотографії усіх об’єктів, з виділенням характерних точок, для кожного образу на знімках із супутників, за якими можливо ідентифікувати об’єкт.

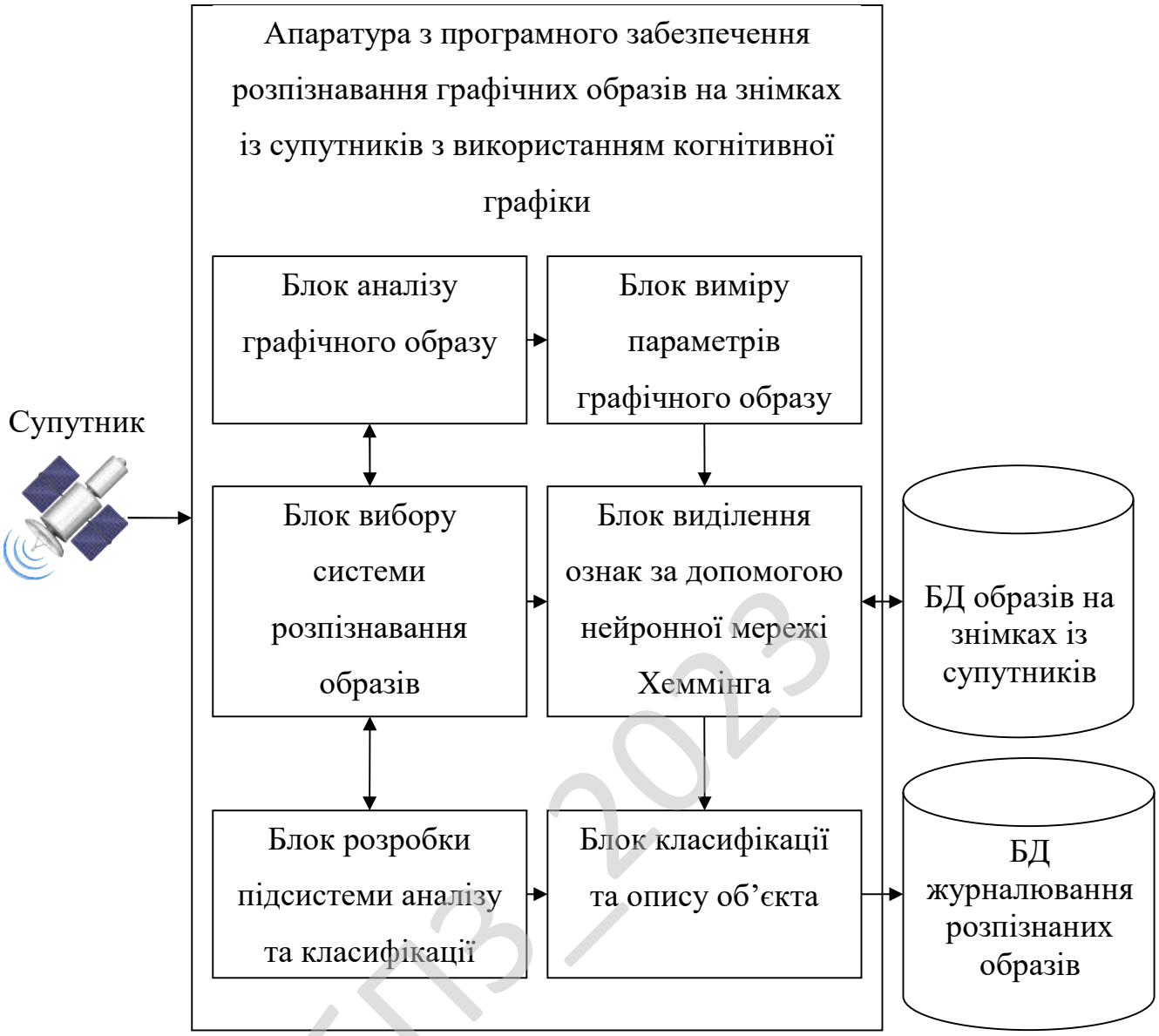

Рисунок 3.1 – Структурна схема системи

3. Камера на супутнику, з якої поступає інформація систему розпізнання образів.

4. Система розпізнання образів, яка складається з наступних блоків:

– Блок читання картинки з камери на супутнику. Він призначений для читання картинок з камери на супутнику й подання даних на блок аналізу та виділення ознак за допомогою нейронної мережі Хеммінга.

– Блок аналізу та виділення ознак за допомогою нейронної мережі Хеммінга. Він є основою системи, й за допомогою нижчеописаного алгоритму

проводить розпізнання осіб, та машин, які перетнули межу території установи.

– Блок класифікації та опису об’єкта. Він дозволяє, виходячи з даних, отриманих від блоку аналізу та виділення ознак за допомогою нейронної мережі Хеммінга, розподілити куди заносити отримані дані, у поля бази даних, які відповідають за осіб, або у поля бази даних, які відповідають за машини.

Однак завдання розпізнавання, саме по собі, припускає інтелектуальну обробку отриманої інформації, що представляє певні складності. Але яких-небудь універсальних методів обробки зображення, порівнянних по продуктивності і якості розпізнавання з людськими здатностями, немає. Наприклад, у завданнях, які відносяться перед експертними системами, потрібно більше глибокий інтелектуальний аналіз і високу швидкодію, цими ж властивостями повинні володіти роботизовані системи обслуговування. Тому обробка зображення в завданні розпізнавання є однією із центральних проблем.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема системи включає в себе наступні функціональні блоки:

1. Блок розпізнання образів.
2. Блок розпізнавання образів на знімках із супутників.
3. Блок динамічного спостереження за об’єктом.
4. Блок документування подій на об’єкті.

Розглянемо ці блоки більш детально.

Блок розпізнання образів

Виділимо етапи при рішенні завдання розпізнавання зображень:

– Сприйняття поля зору.

– Сегментація.

– Нормалізація виділених об'єктів.

– Розпізнавання.

Виходячи із цього, використовуються наступні основні принципи:

– Принцип цілісності – розпізнаваний об'єкт розглядається як єдине ціле, що складається зі структурних частин, зв'язаних між собою просторовими відносинами.

– Принцип двонаправленості – створення моделі ведеться від зображення до моделі й від моделі до зображення.

– Принцип передбачення. Полягає у формуванні гіпотези про зміст зображення.

– Принцип цілеспрямованості, що включає сегментацію зображення й спільну інтерпретацію його частин.

– Нічого не робити без процедури розуміння (сприйняття поля зору).

– Принцип максимального використання моделі проблемного середовища, використання заздалегідь відомих, апріорних параметрів.

Етап інтерпретації зображення не позначений чіткими границями й включається частково в процес сегментації й остаточно завершується на етапі розпізнавання.

Природно задатися питанням: а чи не можна брати зображення й послідовно порівнювати його з еталонами по ряду яких-небудь ознак? Але отут виникає ряд проблем і складностей:

– Тло. Як правило, зображення пред'являються на складному динамічному тлі.

– Орієнтація. Зображення еталона й вхідних зображень відрізняються положенням у поле зору.

– Перешкоди. Вхідні зображення не збігаються з еталонами за рахунок випадкових і локальних перешкод.

– Висвітлення. Відмінності вхідних і еталонних зображень виникає за рахунок зміни освітленості, підсвічування.

– Перетворення. Еталони й зображення можуть відрізняти складні геометричні перетворення.

Блок розпізнавання образів на знімках із супутників

Система захвата осіб дозволяє виділяти тільки особи образів на знімках із супутників, вибирати найбільш виразне зображення з декількох варіантів і зберігати їх у базі даних. Далі система розпізнавання образів на знімках із супутників ідентифікує образи і автоматизує пошук зображень у базах даних.

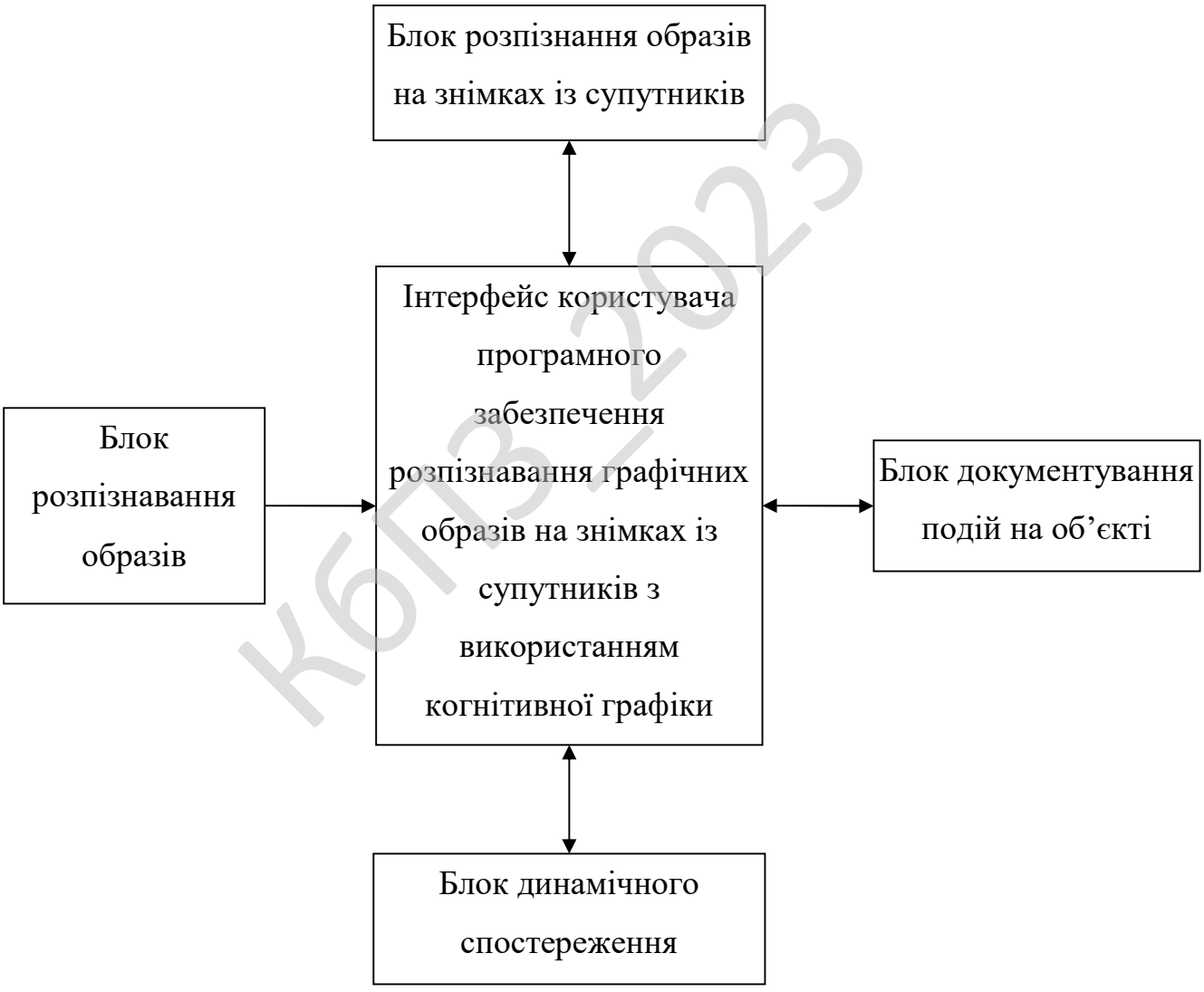

Рисунок 3.2 – Функціональна схема системи

Блок динамічного спостереження за об’єктом

Системи динамічної цілевказівки аналізують зміни координат характерних точок об'єкта, наприклад центра ваги, кольору.

Блок документування подій на об’єкті

Матеріал відеоархівів може виявитися корисним як доказова база при розслідуванні несанкціонованих дій.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Виведення вікна ПЗ.

– Моніторинг БД.

– Редагування БД.

– Формування та додавання інформації до файлів.

– Додавання графічних образів.

– Розпізнавання образів з використанням когнітивної графіки.

– Завантаження графічних образів.

– Виведення результатів розпізнавання.

– Занесення результатів розпізнавання у БД.

– Навчання нейронної мережі.

– Вибір даних для навчання з БД.

– Вибір величини похибки розпізнавання.

– Виведення результатів навчання.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

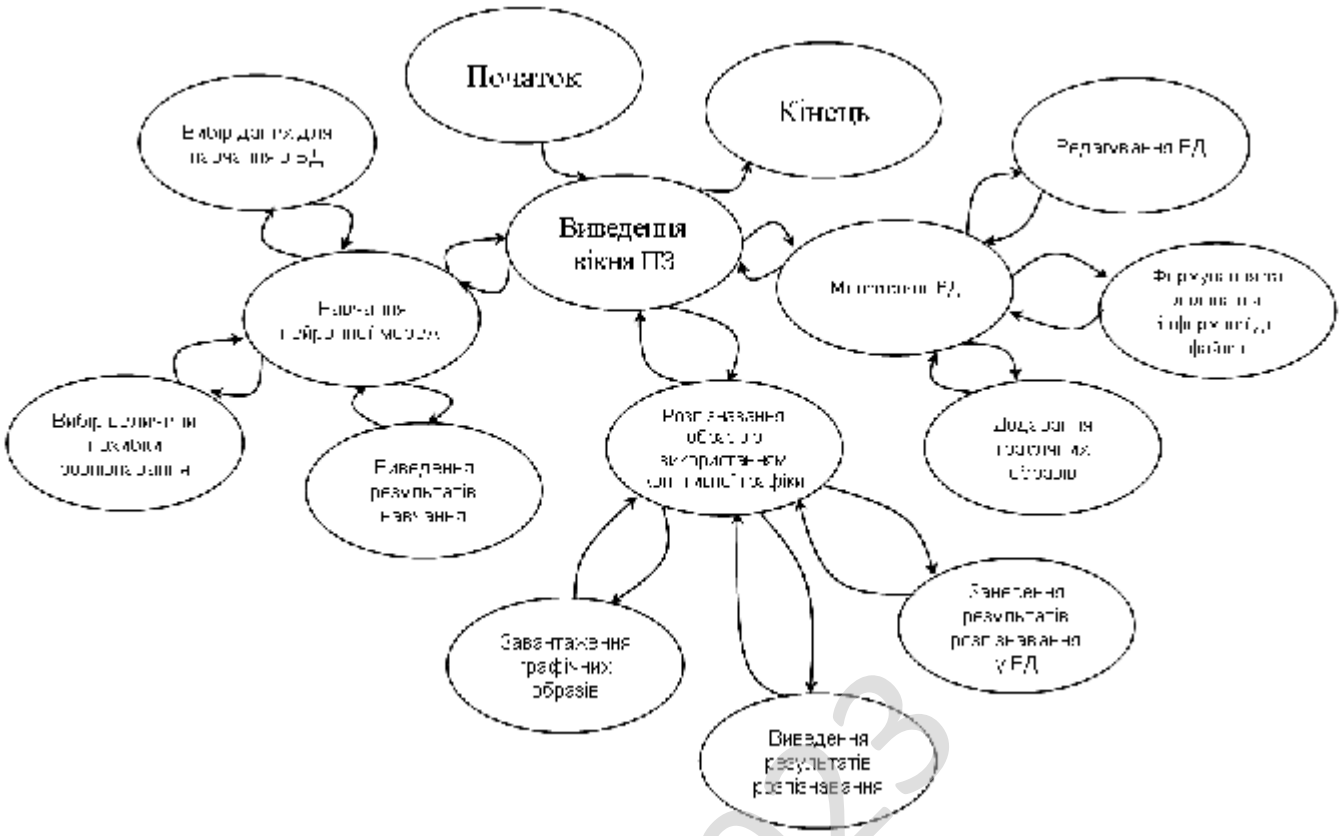

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

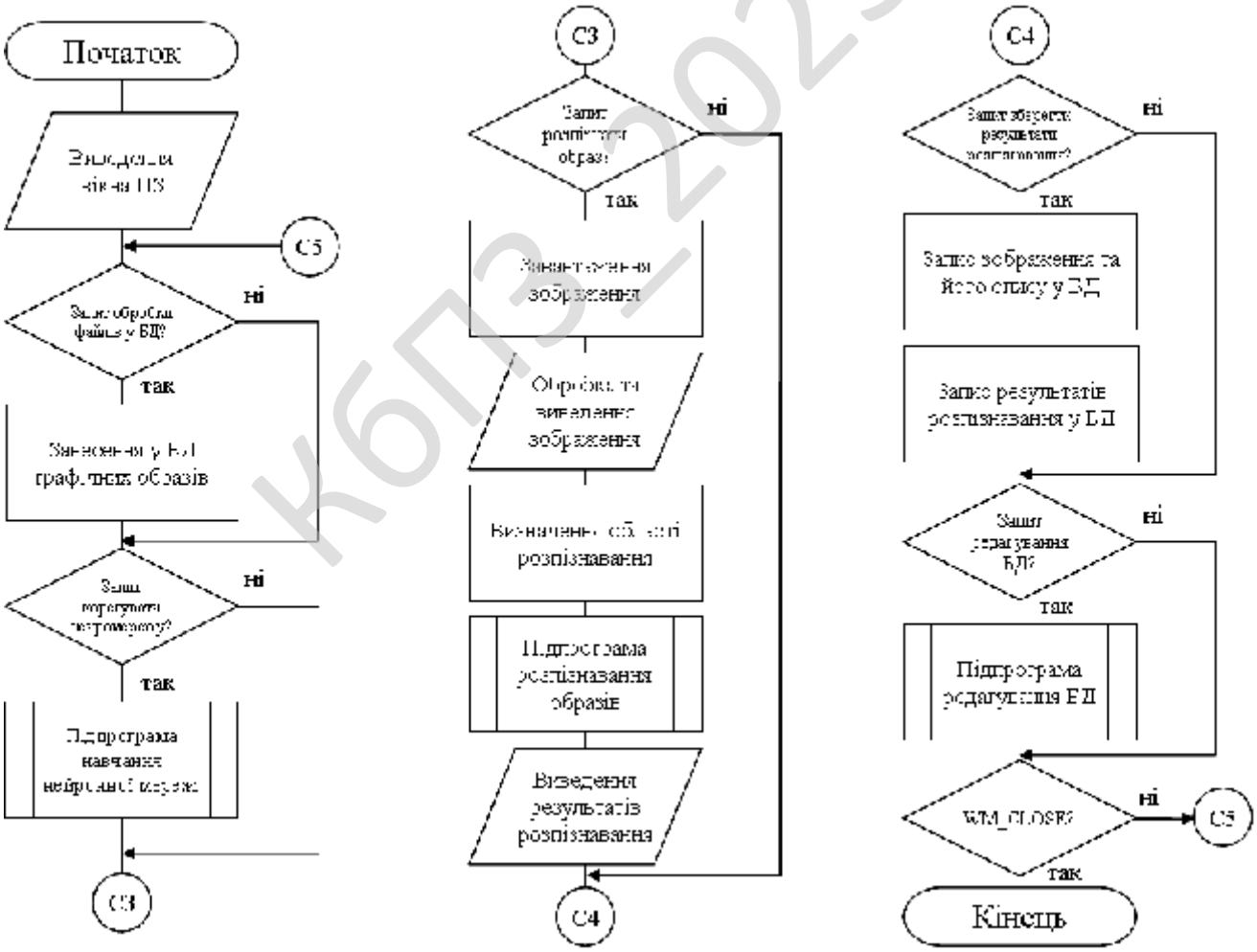

Рисунок 4.1 – Блок схема основної програми

При виборі початкової точки відліку при побудові схем я враховував, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулям системи.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограм та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Підпрограма ініціалізації мережі:

```
// Ініціалізація мережі значеннями з таблиці
procedure TForm1.Init;
begin
  // Очистити мережу від зразків
  NeuralNetHem.ResetPatterns;
  // Додати зразки з таблиці до мережі
  Table.First;
  while not Table.Eof do
  begin
    AddPattern(TableLETTERS.AsString);
    Table.Next;
  end;
  Table.First;
  // Ініціалізувати ваги
  NeuralNetHem.InitWeights;
```

```
  // Мережа підготовлена до розпізнавання
end;
```

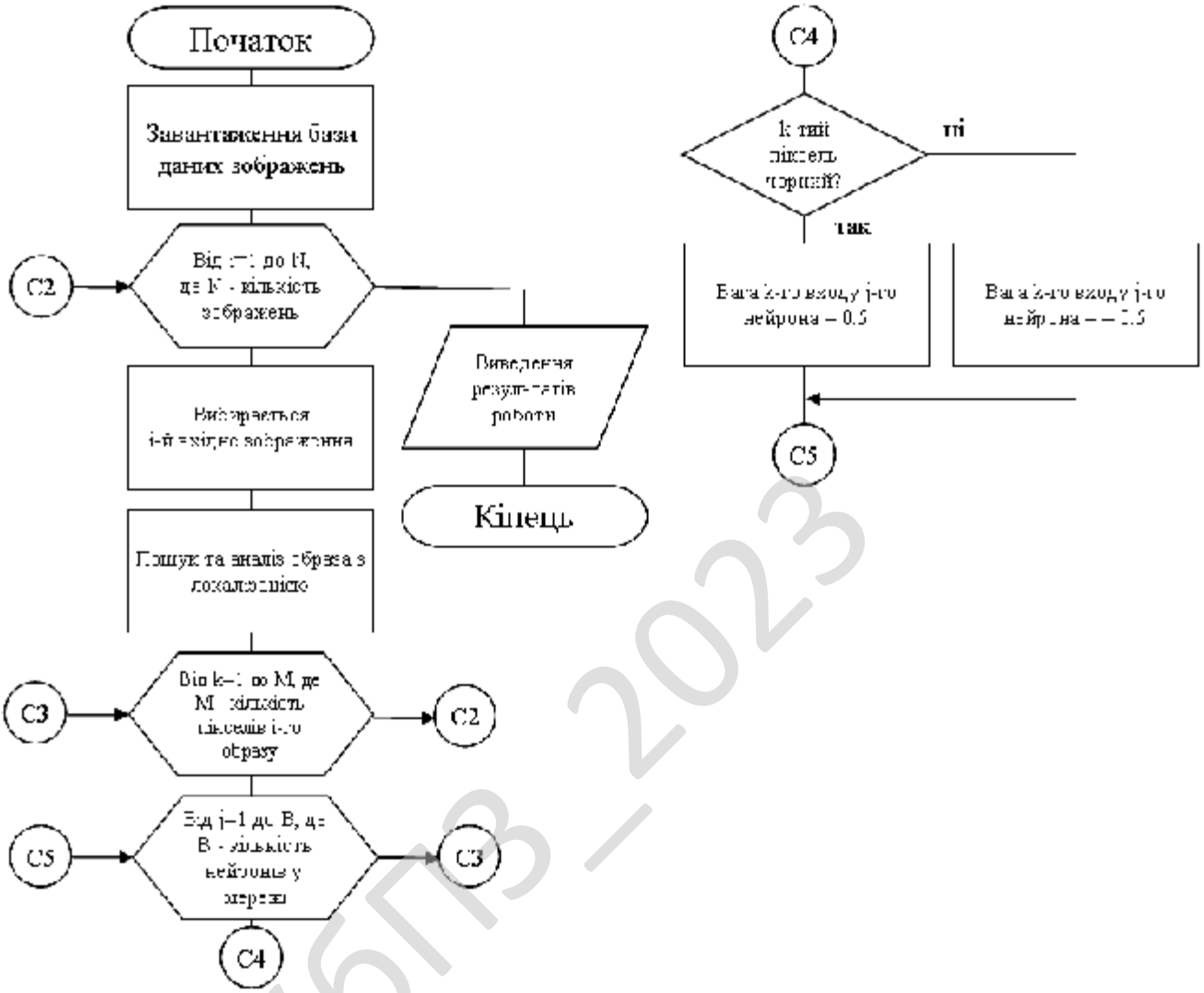

Рисунок 4.2 – Блок схема підпрограми

Розглянемо підпрограму навчання нейронної мережі:

```
procedure TForm1.N;
var
  i, j, k: integer;
begin
  if FUseForTeach20152015 = 100 then
  begin
    PatternCount := FFields[0].DataInCount;
    TestSetPatternCount := 0;
  end
```

```
          else
          begin
PatternCount := Round((FFields[0].DataInCount - 1) * FUseForTeach20152015 / 100);
            TestSetPatternCount := FFields[0].DataInCount - PatternCount;
          end;
          if not Teach20152015Stopped then
            NormalizeData;
        // формування вхідних значень навчальної множини
          RealOutputIndexCount := OutputFieldCount;
          RealInputIndexCount := InputFieldCount;
          k := 0;
          for i := 0 to FAvailableFieldsCount - 1 do
            if FFields[i].KindName = fdInput then
            begin
              for j := 0 to PatternCount - 1 do
                FPatternsInput[j, k] := FFields[i].DataIn[j];
              // запам'ятовує індекс поля
              RealInputIndex[k] := i;
              Inc(k);
            end;
          // формування вихідних значень навчальної множини
          k := 0;
          for i := 0 to FAvailableFieldsCount - 1 do
            if FFields[i].KindName = fdOutput then
            begin
              for j := 0 to PatternCount - 1 do
                FPatternsOutput[j, k] := FFields[i].DataIn[j];
              // запам'ятовує індекс поля
              RealOutputIndex[k] := i;
              Inc(k);
           end;
        // формування вхідних значень тестової множини
          k := 0;
          for i := 0 to FAvailableFieldsCount - 1 do
            if FFields[i].KindName = fdInput then
            begin
              for j := PatternCount to FFields[i].DataInCount - 1 do
                FTestSetPatterns[j - PatternCount, k] := FFields[i].DataIn[j];
              Inc(k);
            end;
        // формування вихідних значень тестової множини
          k := 0;
```

```
  for i := 0 to FAvailableFieldsCount - 1 do
    if FFields[i].KindName = fdOutput then
    begin
      for j := PatternCount to FFields[i].DataInCount - 1 do
        FTestSetPatternsOut[j - PatternCount, k] := FFields[i].DataIn[j];
      Inc(k);
    end;
  // навчання або донавчання мережі
  Teach20152015OffLine;
end;
procedure TNeuralNetBP.Teach20152015OffLine;
var
  j: integer;
  xQuadError: double;
  xNewEpoch: boolean;
begin
  DoOnBeforeTeach20152015;
  if not ContinueTeach20152015 then
  begin
  // ваги ініціалізуються, якщо мережа навчається з початку
    InitWeights;
    FEpochCurrent := 1;
  end;
  Randomize;
  SetLength(FRandomOrder, FPatternCount);
  Teach20152015Stopped := False;
  while (FEpochCurrent <= EpochCount) do
  begin
    FTeach20152015Error := 0;
    FMaxTeach20152015Residual := 0;
    FRecognizedTeach20152015Count := 0;
    xNewEpoch := True;
    Shuffle;
    for j := 0 to PatternCount - 1 do
    begin
      LoadPatternsInput(FRandomOrder[j]);
      LoadPatternsOutput(FRandomOrder[j]);
      Propagate;
      xQuadError := QuadError;
// перевірка - чи розпізнана приклад з навчальної множині
      if xQuadError < IdentError then
        Inc(FRecognizedTeach20152015Count);
```

```
      FTeach2015Error := FTeach2015Error + xQuadError;
  // максимальна помилка на навчальній множині
      if xNewEpoch then
      begin
        FMaxTeach2015Residual := xQuadError;
        xNewEpoch := False;
      end
      else
        if MaxTeach2015Residual < xQuadError then
          FMaxTeach2015Residual := xQuadError;
      CalcLocalError;
      AdjustWeights;
    end;
  // середня помилка на навчальній множині
    FMidTeach2015Residual := Teach2015Error/PatternCount;
  // перевірка мережі на узагальнення
    if TestSetPatternCount > 0 then
      CheckTestSet;
    DoOnEpochPassed;
    if StopTeach2015 then
    begin
      Teach2015Stopped := True;
      Exit;
    end;
   Inc(FEpochCurrent);
  end;
  DoOnAfterTeach2015;
end;
```

Процедура додавання нового образу до мережі має наступний вигляд:

```
// Додавання нового образу до мережі
procedure TForm1.AddPattern(Value: string);
var
  i: integer;
  xVector: TVectorInt;
begin
  SetLength(xVector, stgDatabase.RowCount * stgDatabase.ColCount);
  // Перетворення вимвольного рядка у вектор
  for i := 1 to stgDatabase.RowCount * stgDatabase.ColCount do
    try
      if TableLETTERS.AsString[i] = '1' then
```

```
        xVector[i - 1] := 1
      else
        xVector[i - 1] := -1;
    except
      xVector[i - 1] := -1;
    end;
  //
  NeuralNetHem.AddPattern(xVector);
end;
```

Процедура розпізнавання образів:

```
// розпізнавання символу
procedure TForm1.btnExecuteClick(Sender: TObject);
var
  i, j: integer;
  xString: string;
begin
  // Подаємо сигнали на вихід мережі
  for i := 0 to stgInput.ColCount - 1 do
    for j := 0 to stgInput.RowCount - 1 do
      if stgInput.Cells[i, j] = '1' then
     NeuralNetHem.Layers[1].Neurons[i * stgInput.RowCount + j].Output := 1
      else
    NeuralNetHem.Layers[1].Neurons[i * stgInput.RowCount + j].Output := -1;
  // Запуск процесу розпізнавання
  NeuralNetHem.Calc;
  // Перетворення виходів мережі до рядка
  xString := '';
  for i := 1 to stgOutput.RowCount * stgOutput.ColCount do
    if NeuralNetHem.Layers[1].Neurons[i - 1].Output = 1 then
      xString := xString + '1'
    else
      xString := xString + ' ';
  // Відобразити результат
  ShowMatrix(stgOutput, xString);
end;
procedure TNeuralNetHem.Calc;
var
  i: integer;
  xCurrentIter: integer;
  xArray: TVectorFloat;
begin
```

```
  SetLength(xArray, InputNeuronCount);
 // Цикл працює поки не стабілізуються виходи
  xCurrentIter := 0;
  repeat
    for i := 0 to InputNeuronCount - 1 do
    begin

// Запам'ятовує попередній Крок ітерації, для
      //  цього використовується нульовий шар
      Layers[SensorLayer].Neurons[i].Output := Layers[1].Neurons[i].Output;
      xArray[i] := Layers[1].Neurons[i].Output;
    end;
    for i := 0 to InputNeuronCount - 1 do
      with Layers[1].Neurons[i] do

 // Розраховується новий стан нейронів і аксонів
        ComputeOut(xArray);
    Inc(xCurrentIter);
  until Stabled or (MaxIterCount = xCurrentIter);
  if Assigned(FOnAfterInit) then
    FOnAfterInit(Self);
  SetLength(xArray, 0);
  xArray := nil;
end;
```

У розробленому програмному забезпеченні для розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки використовується нейромережа Хеммінга. Розглянемо детальніше принципи її реалізації.

Нейронна мережа, що складається із двох шарів, кожний з яких містить число нейронів *M*, рівне числу зберігаємих образів. Нейрони першого шару мають *N* зв'язків, з'єднаних із входами мережі (утворюючими фіктивний нульовий шар). Нейрони другого шару зв'язані між собою негативними зворотними зв'язками.

Єдиний позитивний зворотний зв'язок кожний нейрон має із власним виходом. Нейронні мережі Хеммінга можна використовувати для реалізації асоціативної пам'яті в тих випадках, коли немає необхідності, щоб мережа видавала на виході образ у явному вигляді, а досить тільки його номера

(або коду). У порівнянні з мережею Хопфілда мережа Хеммінга має менші витрати на пам'ять і обсяг необхідних обчислень. Принцип роботи мережі Хеммінга заснований на обчисленні відстані Хеммінга від вхідного образу до всіх образів, збережених мережею.

Під відстанню Хеммінга розуміється число різних бітів у двох бінарних векторах. Мережа повинна вибрати образ із мінімальною Хеммінговою відстанню до вхідного. При цьому на виході мережі формується не сам цей образ, а виконується активізація виходу, асоційованого з ним. У мережі Хеммінга два шари – перший і другий шари складаються з *m* нейронів і *m* дорівнює числу зразків. Нейрони першого шару мають по *n* вхідних синапсов, де *n* – розмірність вхідних векторів. Нейрони другого шару зв'язані між собою зворотними, негативними зв'язками.

Зворотний зв'язок від аксона на власника нейрона дорівнює +1. Суть роботи полягає в знаходженні відстані Хеммінга від зразку до всіх зразків. Відстанню Хеммінга називається число різних бітів у двох бінарних векторах.

Мережа повинна вибрати зразок з мінімальною відстанню Хеммінга до поданого вхідного сигналу – у результаті активується один вихід, відповідальний за даний еталонний зразок.

Алгоритм роботи мережі Хеммінга наступний:

1. На входи мережі подається невідомий вектор $\mathbf{X} = \{x_i : i=0... n-1\}$, виходячи з якого розраховуються стани нейронів першого шару (верхній індекс у дужках вказує номер шару):

$$y_j^{(1)} = s_j^{(1)} = \sum_{i=0}^{n-1} w_{ij} x_i + T_j \text{ , } j=0...m-1.$$

2. Після цього отриманими значеннями ініціалізуються значення аксонів другого шару:

$$y_j^{(2)} = y_j^{(1)}, j = 0...m-1.$$

3. Обчислюються нові стани нейронів другого шару:

$$s_j^{(2)}(p+1) = y_j(p) - \varepsilon \sum_{k=0}^{m-1} y_k^{(2)}(p), k \neq j, j = 0...m-1 \text{ ,}$$

і значення їхніх аксонів:

$$y_j^{(2)}(p+1) = f\left[s_j^{(2)}(p+1)\right], j = 0...m-1$$

Активаційна функція f має вигляд порога, причому величина F повинна бути досить великою, щоб будь-які можливі значення аргументу не приводили до насичення.

4. Перевірка, чи змінилися виходи нейронів другого шару за останню ітерацію. Якщо так – перейди до кроку 3. Інакше – завершення роботи.

З оцінки алгоритму видно, що роль першого шару нейронів досить умовна: скориставшись один раз на кроці 1 значеннями його вагових коефіцієнтів, мережа більше не звертається до нього, тому перший шар може бути взагалі виключений з мережі (просто замінений на матрицю вагових коефіцієнтів).

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою IDEA – симетричний блоковий алгоритм шифрування даних, запатентований швейцарською фірмою Ascom. Відомий тим, що застосовувався в пакеті програм шифрування PGP. У листопаді 2000 року IDEA був представлений як кандидат у проекті NESSIE в рамках програми Європейської комісії IST (англ. Information Societes Technology, інформаційні громадські технології).

Першу версію алгоритму розробили в 1990році Лай Сюецзя (Xuejia Lai) і Джеймс Мессі (James Massey) зі Швейцарського інституту ETH Zürich (за контрактом з Hasler Foundation, яка пізніше влилася в Ascom-Tech AG) як заміна DES (англ. Data Encryption Standard, стандарт шифрування даних) і назвали її PES (англ. Proposed Encryption Standard, запропонований стандарт шифрування). Потім, після публікації робіт Біхамом і Шаміра по диференціальному криптоаналізу PES, алгоритм був поліпшений з метою посилення криптостійкості і названий IPES (англ. Improved Proposed Encryption Standard,

покращений запропонований стандарт шифрування). Через рік його перейменували в IDEA (англ. International Data Encryption Algorythm).

Так як IDEA використовує 128-бітний ключ і 64-бітний розмір блоку, відкритий текст розбивається на блоки по 64 біт. Якщо таке розбиття неможливо, останній блок доповнюється різними способами певною послідовністю біт. Для уникнення витоку інформації про кожному окремому блоці використовуються різні режими шифрування. Кожен вихідний незашифрований 64 – біт ний блок ділиться на чотири підблока по 16 біт кожен, так як всі алгебраїчні операції, що використовуються в процесі шифрування, відбуваються над 16-бітними числами. Для шифрування і розшифрування IDEA використовує один і той же алгоритм.

Позначення операцій:

- $\boxplus$ Додавання за модулем 2^{16}.
- $\odot$ Множення за модулем $2^{16}+1$.
- $\oplus$ Побітова виключна диз'юнкція.

Фундаментальним нововведенням в алгоритмі є використання операцій з різних алгебраїчних груп, а саме:

Додавання за модулем 2^{16}.

Множення за модулем $2^{16}+1$.

Побітова виключна диз'юнкція (XOR).

Ці три операції несумісні в тому сенсі, що ніякі дві з них не задовольняють дистрибутивному закону, тобто:

$$a \odot (b \oplus c) \neq (a \odot b) \oplus (a \odot c).$$

Застосування цих трьох операцій ускладнює криптоаналіз IDEA в порівнянні з DES, який базується виключно на операції виключає АБО, а також дозволяє відмовитися від використання S-блоків і таблиць заміни. IDEA є модифікацією мережі Фейстеля.

Генерація ключів

З 128-бітного ключа для кожного з восьми раундів шифрування генерується по шість 16-бітних подключів, а для вихідного перетворення

генерується чотири 16-бітних підключа. Всього буде потрібно 52 = 8 x 6 + 4 різних подключів по 16 біт кожен. Процес генерації п'ятдесяти двох 16-бітних ключів полягає в наступному:

Насамперед, 128-бітний ключ розбивається на вісім 16-бітних блоків. Це будуть перші вісім подключів по 16 біт кожен – $(K_1^{(1)}K_2^{(1)}K_3^{(1)}K_4^{(1)}K_5^{(1)}K_6^{(1)}K_1^{(2)}K_2^{(2)})$

Потім цей 128-бітний ключ циклічно зсувається вліво на 25 позицій, після чого новий 128-бітний блок знову розбивається на вісім 16-бітних блоків. Це вже наступні вісім подключів по 16 біт кожен – $(K_3^{(2)}K_4^{(2)}K_5^{(2)}K_6^{(2)}K_1^{(3)}K_2^{(3)}K_3^{(3)}K_4^{(3)})$

Процедура циклічного зсуву і розбивки на блоки триває до тих пір, поки не будуть згенеровані всі 52 16-бітних підключа.

Шифрування

Структура алгоритму IDEA показана на рисунку 4.3.

Процес шифрування складається з восьми однакових раундів шифрування і одного вихідного перетворення. Вихідний незашифрований текст ділиться на блоки по 64 біта. Кожен такий блок ділиться на чотири підблока по 16 біт кожен. На рисунку ці підблоки позначені D_1, D_2, D_3, D_4. У кожному раунді використовуються свої підключі згідно з таблицею підключів. Над 16-бітними підключами і підблока незашифрованого тексту проводяться наступні операції:

– Множення за модулем $2^{16}+1 = 65537$, причому замість нуля використовується 2^{16}.

– Додавання за модулем 2^{16}.

– Побітове виключне АБО.

В кінці кожного раунду шифрування є чотири 16-бітних підблоки, які потім використовуються як вхідні підблоки для наступного раунду шифрування. Вихідна перетворення являє собою скорочений раунд, а саме, чотири 16-бітних підблоки на виході восьмого раунду і чотири відповідних підключа піддаються операціям:

– Множення за модулем $2^{16}+1$.

– Додавання за модулем 2^{16}.

Після виконання вихідного перетворення конкатенація підблоків D_1', D_2', D_3' і D_4' являє собою зашифрований текст. Потім береться наступний 64-бітний блок незашифрованого тексту і алгоритм шифрування повторюється. Так продовжується до тих пір, поки не зашифрують всі 64-бітові блоки вихідного тексту.

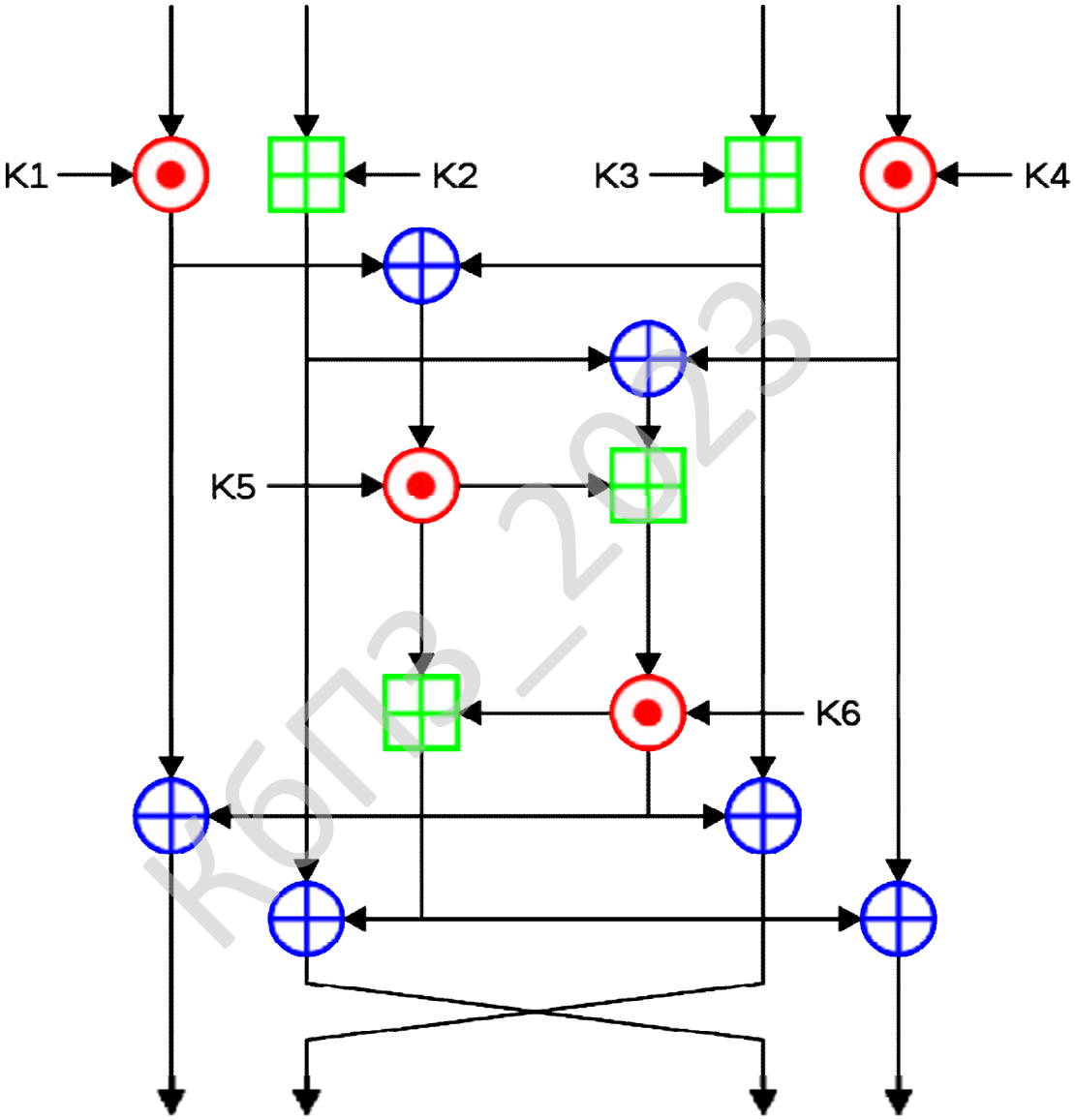

Рисунок 4.3 – Структура алгоритму IDEA

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

– Навчання НМ. З функціональними кнопками: Навчання; Додати до БД; Видалити з БД.

– Розпізнавання образів з основним функціоналом.

– Налаштування.

– Опції.

– Довідка.

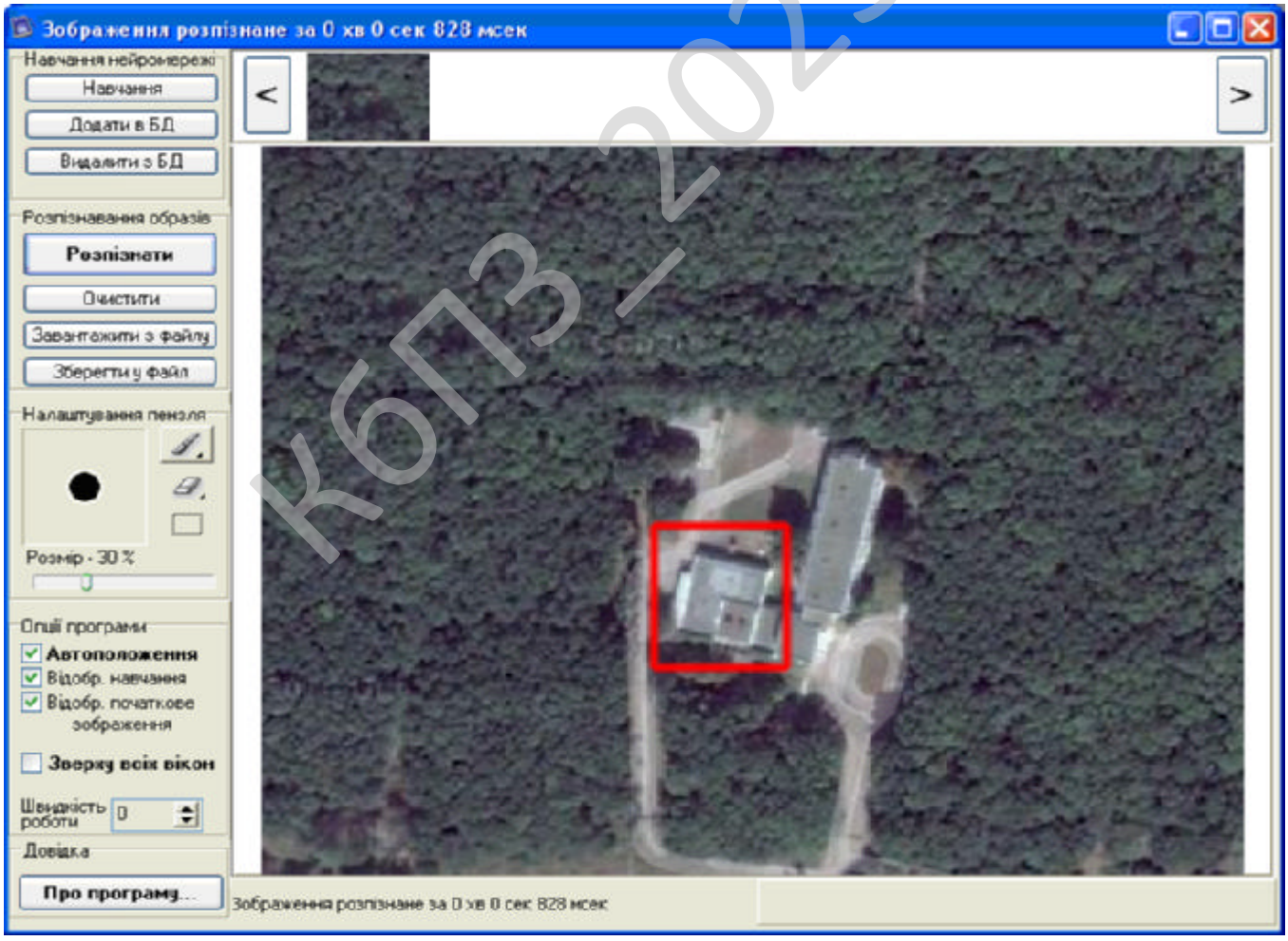

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображене вікно довідки про програму, де можна переглянути короткі відомості про розроблене програмне забезпечення.

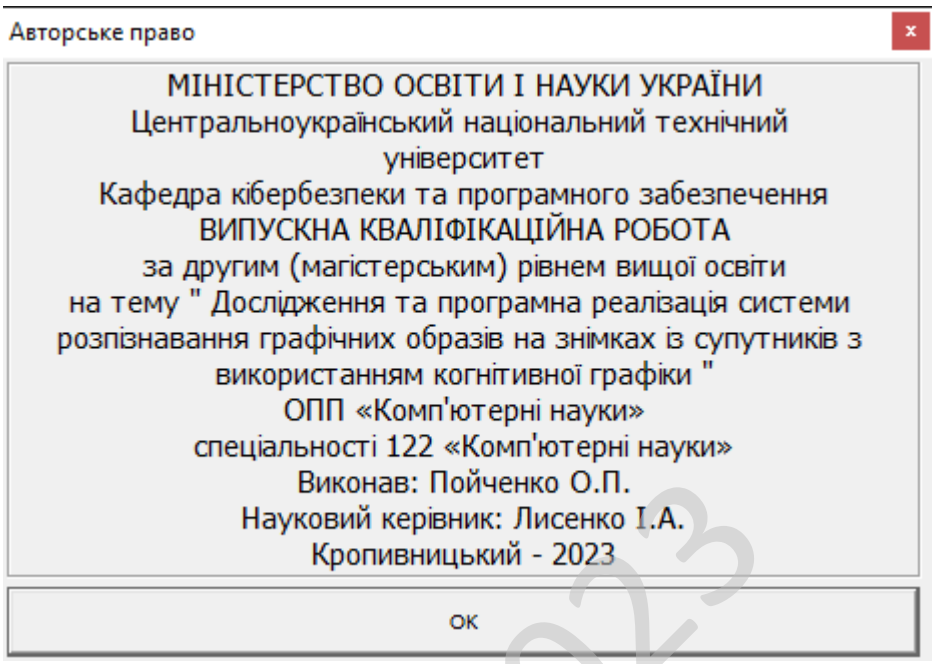

Рисунок 5.2 – Вікно авторського права

Обрано умови розповсюдження – proprietary software. Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж.

Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень.

Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію. Таке програмне забезпечення називається закритим.

Проте, код власницького продукту може бути і відкритим, але власник може обмежити права користувача умовами користувацької ліцензії.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення.

На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт.

Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами. Визначенням власницького програмного забезпечення є не невідповідність хоча б одній з базових умов вільного програмного забезпечення. Сама назва власницьке ПЗ підкреслює визначальне значення власника у способі використання і можливостях розвитку цього програмного забезпечення.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Метою розробки є дослідження та програмна реалізація системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Об'єктом дослідження є процес розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Предметом дослідження є методи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Методи дослідження базуються на методах комп'ютерної графіки, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

– Розроблено вітчизняний продукт розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки, 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 розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

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	120
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

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
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 таблиці 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.	–	120000
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	50
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.:

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

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

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

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

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

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

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

$$T_{уточн} = T_{ном}\, ПV_j, \quad (7.3)$$

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

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

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:

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

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)%.

$$T_{РП} = 0{,}3\cdot 2{,}66\cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot 47 = 79 \text{ люд/день.}$$

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	79	Ф 7.1-7.4
Впровадження	13	Д13
Всього	120	–

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: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{120 \cdot 1}{60 - 5} = 2,2 \text{ ставки.}$$

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	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	39,49

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф^{с}_{др} = \frac{40 \cdot 3}{1,2} = 100 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 100/(60 \cdot 8) = 0{,}2 \text{ ставки.}$$

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	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	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 до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть 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	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	15974	47922
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	2,2	15000	99000
Інженер-електронщик	0,2	12000	7200
Інженер-системотехнік	0,25	12000	9000
Адміністратор мережі	0,5	12000	18000
Системний програміст	0,25	12000	9000
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	12000	9000
Бухгалтер-економіст	0,5	13000	19500
Всього за період розробки	$R_{сп}$ = 5,65	-	$Ф_{роб}$ = 236622

Розрахуємо середньоденну зарплату одного виконавця:

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

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{236622}{5,65 \cdot 60} = 698 \; грн.$$

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:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 500…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:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

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

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 по прайсу фірми Brain за 26.10.23 – джерело http: //brain.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Pentium G6405 (BX80701G6405) 1200, 2 ядра, 4 потоки, 4.1 GHz, TDP - 58 Вт, 14nm box	
Системна плата	GIGABYTE H510M K V2 DDR4, LAN - 1 Гбіт/с, HDMI, 1 x M.2 2280, 2 x Sata 6.0 Gb/s, Micro-ATX	
Відеокарта	Intel UHD Graphics 610	
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) DDR4, 8 ГБ, В наборі - 1	
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 розширення - 7, Додатково - Front 2x12 см, Back 1x12 см, Side 2x12 см	

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
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	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1, 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

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	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

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 7.8 – 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	199177	-	-
Всього по групі	199177	50	99588,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	0	20	0,0
6. Господарський інвентар	28000	25	7000
Всього по групі	33190	-	8297,5
7. Нематеріальні активи	120000	10	12000
Разом	$К_р$ = 1760367		$А_р$ = 190286

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: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 698 \cdot 120/120 = 698 \text{ грн.}$$

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, %.

$$З_д = 698 \cdot 10 \cdot 0{,}01 = 70 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ 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, %.

$$С_{оц} = 0{,}01 \cdot 22(698+70) = 288 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 698 \cdot 15 \cdot 0{,}01 = 105 \text{ грн.}$$

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 приймаємо 1/6 пачки паперу на місяць розробки ($n_р$=1/6). Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 210 грн., визначаємо вартість паперу за період розробки $N_м$ = 3 міс:

$$З_{М1} = Ц_п \cdot n_р \cdot N_м. \quad (7.16)$$

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

Згідно виданих норм до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків в кількості 50 примірників:

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 28,8 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 28,8 грн./шт.

$$З_{М2} = 28,8 \cdot 50 = 1440 \text{ грн.}$$

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.

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

$$З_М = (105 + 1440 + 1702)/120 = 27 \text{ грн.}$$

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

$$О_п = З_о \cdot Н_п \cdot 0,01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 698 \cdot 15 \cdot 0,01 = 105 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 120 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 190286 \cdot 3/(120 \cdot 12) = 396 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$С_п = 698+70+288+105+27+105+396 = 1689 \text{ грн.}$$

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	$З_о$	698
2. Додаткова зарплата виконавців	$З_д$	70
3. Відрахування на соціальні потреби	$С_{оц}$	288
4. Загальногосподарські витрати	$Г_{осп}$	105
5. Витрати на матеріали	$З_М$	27
6. Освоєння нових операційних систем, мов програмування	$О_п$	105
7. Амортизація основних фондів	$А_м$	396
8. Повна собівартість програмного забезпечення	$С_п$	1689
9. Плановий прибуток	$П_р$	845
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	2534
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	506,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3040,8

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 1689 = 845 \text{ грн.}$$

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 7.9.

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

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	3041
Всього капітальних витрат	–	3041

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 обслуговування системи	$З_р$	40260	20130
2. Витрати на електроенергію	$З_{ел}$	621	311
3. Витрати на амортизацію	$З_{ам}$	0	760
Всього витрат за рік	$І$	40881	21201

Витрати на обслуговування роботи системи:

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

де: $Т_р$ – кількість годин обслуговування за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість годин на обслуговування системи зменшилось з 300 год до 150 год на рік.

$$З_{р\,баз} = 300 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 40260 \text{ грн},$$

$$З_{р\,нов} = 150 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 20130 \text{ грн}.$$

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	–	3041	–	760,25
Всього відрахувань	-	–	3041	–	760,25

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($T_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot T_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = 0{,}545 \cdot 300 \cdot 3{,}8 = 621 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}545 \cdot 150 \cdot 3{,}8 = 311 \text{ грн.}$$

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.

$$Ев = (2534 - 1689) \cdot 120 - (0{,}05 \cdot 1408000 + 0{,}4 \cdot 199177 +$$
$$+0{,}25 \cdot 33190 + 0{,}1 \cdot 120000) \cdot 3/12 = 53828{,}5 \text{ грн.}$$

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.

$$T_в = \frac{1760367}{(2534 - 1689) \cdot 120 \cdot 12 / 3} = 4 \text{ роки}$$

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; $К_б$, $К_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (40881\text{-}21201)\text{-}0{,}25 \cdot 3041 = 18920 \text{ грн.}$$

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.	120
2. Повна собівартість розробленої програми	Грн.	1689
3. Ціна розробленої програми	Грн.	2534
4. Плановий прибуток від реалізації розробленої програми	Грн.	845
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1760367
7. Загальний прибуток від реалізації програмної продукції	Грн.	101400
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	53828,5
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Роки	4
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3041
11.Величина економічного ефекту у користувача програмної продукції	Грн.	18920
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,15

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

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{3041}{40881 - 21201} = 0{,}15 \ року .$$

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

Законом України “Про охорону праці” [3] 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, 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», яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007- 98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників 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.

Керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18 розглянемо шкідливі чинники роботи персоналу.

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

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Програміст працює з електронно-обчислювальною машиною (ЕОМ) та іншим обладнанням, яке є джерелом небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. Так як програміст постійно перебуває в приміщенні, тому для комфортних умов праці в цьому приміщенні необхідно створити належний мікроклімат.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– електромагнітні (у т.ч. високочастотні) випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м'язовий апарат;

– недостатня, або надмірна освітленість робочого місця;

– монотонність праці.

Результати досліджень показали, що найбільшою мірою негативний фізіологічний вплив на операторів ПК пов'язано з дискомфортними зоровими умовами через неправильно спроектованого освітлення: пряма і відбита від екранів бляклість, несприятливий розподіл яскравості в полі зору, невірна орієнтація робочого місця щодо світлоприймачів.

Розташовувати обладнане дисплеєм робоче місце необхідно таким чином, щоб в поле зору оператора не потрапляли вікна або освітлювальні прилади. Вони не повинні перебувати і безпосередньо за спиною оператора. Слід домагатися зменшення відображень на екрані від різних джерел штучного і денного світла. Коли штучне світло змішується з природним, рекомендується використовувати лампи, по спектрального складу найбільш близькі до сонячного світла. Співвідношення яскравості екрана і безпосередніх найближчого оточення не повинно перевищувати 3: 1.

Оптимальні значення температури повітря в приміщенні повинні бути 19-23 С. Швидкість руху повітря не більше 0,1 м / с. Рекомендована відносна вологість повітря 55%.

8.3 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 ланцюга).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами, організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.4 Пожежна безпека

Вимоги до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому покладається на посадових осіб за відповідним рішенням керівництва і прописується в посадових інструкціях і положеннях по структурним підрозділам.

Зокрема, вказуються конкретні території, ділянки, зони, об'єкти, цілі будівлі і їх частини, поверхи, на яких відповідального співробітника повинне проводити такі організаційні роботи.

Відповідальні особи зобов'язуються розробити, впровадити та підтримувати в певному інструкцією і положенням на ввірених їм об'єктах

протипожежний режим і інструкції відповідно до вимог, викладених в нормативних актах.

Передбачено також створення підрозділу добровільної пожежної охорони та пожежно-рятувальної команди в його складі.

Встановлений режим включає порядки з описом місць спеціального призначення та правила їх користування та утримання, наприклад:

– евакуаційних шляхів;

– так званих «курилок»;

– місць складування продукції та сировини;

– стоянки транспорту.

Також встановлюється порядок роботи та технічного обслуговування:

– вентиляційного устаткування;

– засобів пожежогасіння і захисту від загорянь;

– нагрівальних приладів;

– електрообладнання.

Розробляються і впроваджуються правила роботи з відкритим вогнем і горючими матеріалами. Створюються графіки проходження інструктажів з пожежної безпеки співробітників, а також порядок і терміни перевірок знань пожежно-технічного мінімуму, в тому числі, тих працівників, які відповідальні за цю ділянку роботи на підприємстві. При цьому можуть передбачатися внутрішні лекції, семінари, тренінги та практичні заняття на підприємстві, а також зовнішні – на базі спеціалізованих навчальних центрів з професійними викладачами.

Важливою складовою протипожежного режиму на будь-якому об’єкті є розробка і впровадження порядку дій при виникненні пожежі. Неодмінно має бути план евакуації, описано, як повинні відключатися електроустановки, що і в якій послідовності необхідно робити співробітникам.

Відповідно, для кожного об’єкта, кожного приміщення (крім коридорів, санвузлів, басейнів і подібних приміщень), окремих видів робіт складаються

інструкції, за якими повинен працювати персонал, залучений на певних ділянках і в виконанні окремих видів робіт. За інструкціями проводиться навчання (інструктаж) персоналу з подальшим контролем знань.

Детально про те, як розробити протипожежний режим, прописати порядки та інструкції, пояснюють на тематичних курсах і семінарах.

Відповідно ДБН В 1.1-7-2016 «Пожежна безпека об'єктів будівництва» будинок можна віднести до II групи по ступені вогнестійкості й до категорії Д по ступені пожежонебезпеки.

Від розподільного щита по праву й ліву сторони встановлені кондиціонери, зовнішня електропроводка, поміщена в ізольований кабель. Висота проводки становить 2,2 м від рівня підлоги, її кріплення здійснюється за допомогою металевих власників. Біля кожного стола організований розподільний щит, розташований на текстолітовій пластинці, закріпленої на стіні на рівні 1м від підлоги. Усього до складу входять п'ять розеток і дві клеми заземлення. Всі обчислювальні машини з'єднані із клемами заземлення. Чотири з п'яти розеток забезпечують подачу напруги 220 V, а одна, забезпечує подачу напруги в 36 V. Про це є відповідні написи на кожному розподільному щиті.

Робота обслуговуючого персоналу полягає в інсталяції необхідного програмного забезпечення й наступному його використанні в діалоговому режимі роботи з ЕОМ. Іноді може виникати необхідність написання допоміжних програм для поліпшення роботи вузла або для зниження витрат. З погляду забезпечення умов праці й вимог техніки безпеки для роботи програміста необхідно наступне: достатнє висвітлення екрана дисплея й робочого місця; повна технічна справність устаткування, його електробезпечність; достатня пожежна безпечність приміщення; оптимальний мікроклімат, що сприяє продуктивній роботі; відповідність робочого місця вимогам ергономіки. До небезпечних і шкідливих факторів, дії яких піддається програміст, можна віднести: можливість поразки електричним струмом, при електроні справності встаткування, порушенні заземлення або техніки безпеки; робота в мікрокліматі з неприпустимими

параметрами; робота при недостатній освітленості екрана дисплея й робочого місця.

Відповідно НПАОП 40.1-1.21-98 “Правил безпечної експлуатації електроустановок споживачів”, приміщення можна віднести до приміщень без підвищеної небезпеки, оскільки це приміщення, сухе, з нормальною температурою й ізолюючими підлогами, що не має заземлених металоконструкцій.

Персональні ЕОМ можна віднести до першого класу електротехнічних виробів по способі захисту людини від поразки електричним струмом, оскільки їхні корпуси зроблені з ізолюючої пластмаси й кожен пристрій має заземлення. Відповідно правилам пристрою електроустановок ЕОМ можна віднести до електроустановок з робочою напругою до 1000 В.

Однієї з достовірних причин пожежі в приміщенні з обчислювальною технікою може бути коротке замикання, що спричиняє спалах електропроводки. Для його попередження вся обчислювальна техніка, а також інші електричні пристрої повинні бути обладнані плавкими запобіжниками, а на вході електромережі повинен бути передбачений автомат захисту. Не слід користуватися електричними подовжувачами й трійниками, що не мають сертифікатів відповідності вимогам безпеки.

Необхідно передбачити наявність у межах досяжності первинних засобів гасіння пожежі (вогнегасників) для локалізації вогню власними засобами до приїзду команди пожежної охорони. Повинен бути розроблений план екстреної евакуації персоналу при виникненні загоряння. Кількість евакуаційних виходів повинне бути не менш двох. Допускається використання одного евакуаційного виходу, якщо відстань найбільш віддаленого робочого місця до цього виходу не перевищує 25 м.

8.5 Розрахунок занулення

Розрахунок занулення складається з трьох частин:

1. Розрахунок на відключаючу спроможність;

2. Визначення максимальної напруги на корпусі обладнання відносно землі при замиканні фази на корпус;

3. Розрахунок робочого і повторного заземлювачів

Початкові дані

1. Потужність електроприладів, які підлягають зануленню:

P = 7 кВт.

2. Довжина магістрального кабеля: LM=45 м.

3. Довжина розгалудження (від розподільчого щита до електроприладів) : l=25 м.

4. Матеріал провідників кабеля − алюміній.

5. Лінійна напруга U=380 В.

6. Фазна напруга Uф=220 В.

Розрахунок проводиться відповідно до схеми електромережі, яка зображена рис. 8.1.

У результаті розрахунку отримали:

– номінальна силу струму апарата захисту 25 А.

– найменше допустиме по умовам спрацьовування захисту значення сили струмукороткого замикання 50 А.

– площа перерізу магістрального кабеля (провідника) 8 мм$^{2.}$.

– максимальної напруги на корпусі обладнання відносно землі при замиканні фази на корпус 30,5 В.

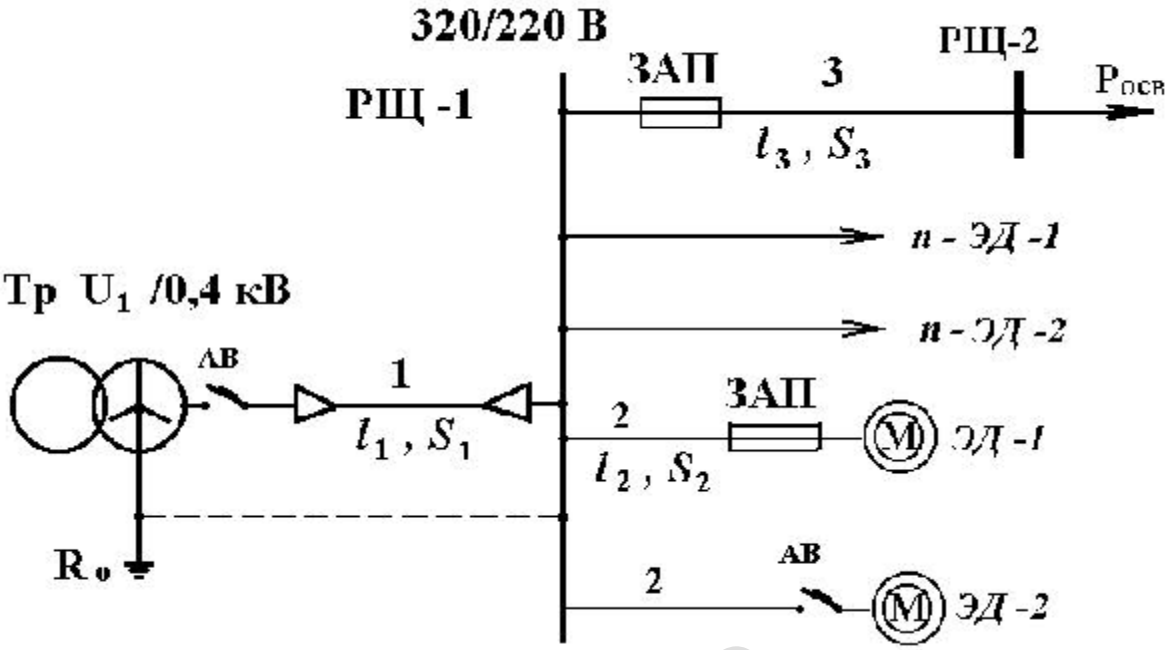

Рисунок 8.1 – Схема електромережі.

Тр – трансформатор; РЩ – розподільчий щит; АВ – автоматичний вимикач; ЗАП – запобіжник; 1 – магістральний кабель; 2 – розгалуження до електроприладів.

Список використаних джерел інформації

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

2. Охорона праці. Ч. 2. Занулення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ−сумісного типу / [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака, А. Е. Солових, С. Е. Катеринич]; Мін-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – 2−ге вид., перероб. та доп. – Кропивницький : ЦНТУ, 2019. – 27 с. URI: http://dspace.kntu.kr.ua/jspui/handle/123456789/8769

3. 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 (дата звернення 19.10.23).

4. 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" / М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програм. забезпечення; [укл. О.В. Оришака, К.М. Марченко]. – Кропивницький: ЦНТУ, 2022. – 19 с. [Електронний ресурс]. – Режим доступу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12240 (дата звернення 19.09.23).

5. НПАОП 40.1-1.21-98. Правила безпечної експлуатації електроустановок споживачів – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0093-98#Text (дата звернення 19.09.23).

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

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 розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

– Досліджена система розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

– На основі отриманих результатів досліджень створена програмна реалізація системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

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 Delphi 10.4.1. 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 IDEA.

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 18920 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,15 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пойченко О.П. Дослідження та програмна реалізація системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Henry Lloyd. Interactive Computer Graphics. States Academic Press. 2022. 247 p.

3. Ranjan Parekh. Fundamentals of Image, Audio, and Video Processing Using MATLAB®With Applications to Pattern Recognition. CRC Press. 2021. 406 p.

4. Alasdair McAndrew. A Computational Introduction to Digital Image Processing. Chapman & Hall. 2021. 560 p.

5. PeterShirley, SteveMarschner. Fundamentals of Computer Graphics. 2009

6. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.

7. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.

8. Інженерна комп’ютерна графіка: підручник / В.В. Проців [та ін.] / М-во освіти і науки України, Нац. гірн. унт-т. – Дніпро: НГУ, 2017. – 247 с.

9. Проців В.В. Прикладна комп'ютерна графіка [Текст]: Навч. посібник / В.В. Проців, К.А. Зіборов, К.М. Бас, Г.К. Ванжа; М-во освіти і наук, Нац. гірн. ун-т. – Д.: НГУ, 2016. – 187 с.

10. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph.. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.

11. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.

12. 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.

13. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022,

14. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

15. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w

16. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

17. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) – 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

18. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

20. 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.

21. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». CEUR Workshop Proceedings Volume 2654, 2020, Pages 1-14.

22. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

23. 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.

24. 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.

25. 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.

26. 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.

27. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», CEUR Workshop Proceedings Volume 2608, 2020, Pages 646-660.

28. Zhurakovskyi, B., Tsopa, N., Batrak, Y., Odarchenko, R., Smirnova, T «Comparative analysis of modern formats of lossy audio compression». Workshop Proceedings, 2020, 2654, стр. 315-327.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.

38. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макрофотографій». IV міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30.

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. 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.

41. 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.

42. О. Смірнов, Є. Деменко, О. Онікійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послідовностей для приховування даних в зображеннях» Комп'ютерні науки та кібербезпека. № 4. С. 30-37. 2019.

43. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

44. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

45. 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

46. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

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

48. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". – Випуск 3 (140). – Х.: ХУПС – 2016. – С. 36-39.

49. 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.

50. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник наукових праць "Системи обробки інформації". – Випуск 2 (118). т.2. – Х.: ХУПС – 2014. – С. 64-67

51. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник тез VI міжнародної науково-практичної конференції "Проблеми та перспективи розвитку IT-індустрії". м. Харків. 17-18 квітня 2014р. – Харків: ХНЄУ. – 2014. – С. 240.

52. Смірнов О.А., Коваленко О.В., Кожанова А.С., Левошко О.Л., Константинова Л.В. Основи системного програмування. Навчальний посібник для студентів вищих навчальних закладів напрямів підготовки 8.050102 «Комп'ютерна інженерія». За ред. Коваленка О.В., Гриф "Навчальний посібник" надано у відповідності з листом Міністерства освіти і науки України від 26.02.2013 року № 1/11-4368. – Кіровоград: КНТУ 2013. – 257с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

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. № 37-13 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 розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки.

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 розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки;

– 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 Delphi 10.4.1.

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 повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

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 – 98 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 15.12.2023 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

_____ Лисенко І.А.

Дослідження та програмна реалізація системи розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 59

Літера: РП

Кропивницький – 2023 року

Файл Main_Cognitive_graphic_.pas - основна програма

```
unit Main_Cognitive_graphic_;

{---------Головний модуль програми керування--------}

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, VCap,VCapStrings ,StdCtrls,DirectShow,
ExtCtrls,ComCtrls,ComObj,Active,Speech,
  UPreview, //модуль із компонентами для захоплення відео
  UGraphConfig, //модуль інтерфейс, що реалізує, налаштування камер
  USingleFrame, //модуль перегляд, що реалізує, окремих кадрів
  UTypeConst, //модуль утримуючі загальні типи й константи
  Can_Dll, //модуль імпортує фукції з Can.dll
  UCommunication, //модуль із фукціями для повідомлення по Кан'у
  URazvertka, //модуль циклічного розгорнення
  UCamTimers, //таймери камер
  UMoving, //руху
  URazvertkaConfig, //форма налаштування розгорнення
  URoboRootServer, //робосервер
  URoboServerConfig, //форма налаштування робосервера
  USystemCoordinat, //система координат
  URisovalka, //рисовалка
  UCommunicationForm, //форма висновку інформації про маяк
  Picture_Analyz_Hamming, //аналіз букв
  UQPixels, //швидкі пікселі
  Picture_processing_Hamming,
  URestaran,
  UDebug,
  UEmul,
  UDialog, BCPort,
  about;

const port = 'COM2';

type
  TMainForm = class(TForm)
    Button3: TButton;
    MoveForLine: TButton;
    CamsListBox: TListBox;
    Label7: TLabel;
    RefreshCamsListButton: TButton;
    CamConfigButton: TButton;
    CamsGroupBox: TGroupBox;
    DialogListBox: TListBox;
    Label1: TLabel;
    VideoModeComboBox: TComboBox;
    Label2: TLabel;
    RisovalkaButton: TButton;
    CommunicationButton: TButton;
    AnalyzFrameButton: TButton;
    RazvertkaButton: TButton;
    OnFlyDebugButton: TButton;
    Button1: TButton;
    Button2: TButton;
    TLabel: TLabel;
    AddSampleButton: TButton;
    RestaranButton: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Label3: TLabel;
    Timer1: TTimer;
    Edit1: TEdit;
```

```
    Edit2: TEdit;
    Label4: TLabel;
    Button7: TButton;
    NeuroCount: TEdit;
    Button8: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure MovingFormButtonClick(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure MoveForLineClick(Sender: TObject);
    procedure RefreshCamsListButtonClick(Sender: TObject);
    procedure CamConfigButtonClick(Sender: TObject);
    procedure CamsListBoxClick(Sender: TObject);
    procedure DialogListBoxDblClick(Sender: TObject);
    procedure VideoModeComboBoxChange(Sender: TObject);
    procedure CommunicationButtonClick(Sender: TObject);
    procedure RisovalkaButtonClick(Sender: TObject);
    procedure AnalyzFrameButtonClick(Sender: TObject);
    procedure RazvertkaButtonClick(Sender: TObject);
    procedure OnFlyDebugButtonClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure AddSampleButtonClick(Sender: TObject);
    procedure RestaranButtonClick(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
    {Для розгорнення}
    //настроїти параметри циклічного розгорнення
    procedure ConfigRazvertka(CamNum: Cardinal);
  end;

function Send:integer; //запис даних у буфер КАН'а для відправлення

type
  mass = array [1..250] of byte;
  reg_array = ^smallint;


function mbConnect(const port: string ; speed: integer;parity: integer;
stopbits: integer;flow: integer): integer;
 cdecl external 'MODBUS.dll' ;
function mbDisconnect(): integer;
 cdecl external 'MODBUS.dll';
function mbExecuteProgramFile(dev: cardinal; filename: string): integer;
 cdecl external 'MODBUS.dll';
function mbReadHoldingRegisters  (dev: cardinal; dest: reg_array; address:
cardinal; count: cardinal): integer;
 cdecl external 'MODBUS.dll';
FUNCTION mbReportDeviceID(dev: CARDINAL;  dest: mass; max_len: cardinal;
actual_len : Pointer): integer;
 cdecl external 'MODBUS.dll';
function mbReset( dev: cardinal): integer;
 cdecl external 'MODBUS.dll';
function mbSetLogDetails(log_errors: boolean;log_messages: boolean;log_data:
boolean): integer;
 cdecl external 'MODBUS.dll';
function mbWriteHoldingRegisters(dev: integer; from: reg_array; address:
cardinal; count: cardinal): integer;
 cdecl external 'MODBUS.dll';
```

```
var
  MainForm: TMainForm; //головна форма

  //масив налаштувань камер
  GraphConfigExArr: TGraphConfigExArr;

  //об'єкт головного потоку робосервера
  RoboRootServerManageThread: TRoboRootServerManagThread;

   //прапор "рух по смузі"
  MoveForLineFlag: boolean = false;

  //камера для висновку відео
  ActiveCamIndex: Integer = -1;

  {для TextToSpeech}
  {Центральний інтерфейс,через який виробляються всі дії з мовою}
  fITTSCentral: ITTSCentral;

  {Інтерфейс для зв'язку з аудіопристроєм}
  fIAMM: IAudioMultimediaDevice;

  {Інтерфейс для перебору движків}
  aTTSEnum: ITTSEnum;

  {Покажчик на параметри движка}
  fpModeInfo: PTTSModeInfo;

  NumFound : DWord;
  ModeInfo : TTSModeInfo;

  QPixels: TQuickPixels;
  timercnt: Cardinal = 0;


   i: integer;
   dest1: mass;


{процедура виводить в ListBox
список доступних діалогів у компоненті VideoCapture}
procedure ShowAvialableDialogs(ListBox: TListBox; VideoCapture: TVideoCapture);

{ функція зберігає відео налаштування у файл зі змінних
У разі удачі повертає true, інакше false}
function SaveGraphConfig(var GraphConfigExArr: TGraphConfigExArr): boolean;

{ функція ініціалізує камери
VideoDeviceList - список всіх доступних камер
у випадку удачі функція повертає true, інакше false}
function InitCams(var GraphConfigExArr: TgraphConfigExArr;
                        VideoDeviceList: TStringList): boolean;

//процедура виводить в ComboBox доступні відео режими на компоненті VideoCapture
procedure ShowVideoModes(ComboBox: TComboBox; VideoCapture: TVideoCapture);

//процедура ініціалізує відео налаштування для камери CamName за замовчуванням
procedure DefaultInitCam(CamName: String; var GraphConfigEx: TGraphConfigEx);

//---------------------------------i


//процедура, що вимовляє текст
procedure SayText(Text: String);

function GetVideoDevicesListEm(): TStringList;
```

```
implementation

uses Math;

{$R *.dfm}

function GetVideoDevicesListEm(): TStringList;
begin
  Result := TStringList.Create;
  Result.Add('Samsung SuperShit Cam');
  Result.Add('Hitachi MegaSlow WebCam');
  Result.Add('Електроніка КХ - 1');
end;


{Допоміжні процедури, що не входять у клас форми}

//процедура, що вимовляє текст
procedure SayText(Text: String);
var
  SData: TSData;
begin
  {Цей текст буде прочитаний}
  SData.dwSize := length(Text) + 1;
  SData.pData := pChar(Text);

  fITTSCentral.TextData(CHARSET_TEXT, 0, SData, nil, IID_ITTSBufNotifySink);
end;
//---------------------------------i

{процедура виводить в ListBox
список доступних діалогів у компоненті VideoCapture}
procedure ShowAvialableDialogs(ListBox: TListBox; VideoCapture: TVideoCapture);
var
  d: TCaptureDialog; //всі можливі ідентифікатори діалогів
begin
  ListBox.Clear; //очищення ListBox
  //додавання в ListBox всіх доступних діалогів
  for d := Low(TCaptureDialog) to High(TCaptureDialog) do
  if d in VideoCapture.Dialogs then
    ListBox.Items.AddObject(DialogTitles[d],TObject(d));

end;
//---------------------------------i

{ функція зберігає відео налаштування у файл зі змінних,
у разі удачі повертає true, інакше false}
function SaveGraphConfig(var GraphConfigExArr: TGraphConfigExArr): boolean;
var
  GraphConfigFile: TextFile; //файлова змінна
  GraphConfigStr: String; //рядок з відео налаштуваннями
  i: integer; //лічильник
begin
  Result := true;

  //відкриття файлу
  AssignFile(GraphConfigFile,GraphConfigFileName);
  {$ I-I-}
  Rewrite(GraphConfigFile);

  for i := 0 to High(GraphConfigExArr) do
  begin
    //одержання налаштувань у строковому форматі
    GraphConfigStr := GraphConfigExArr[i].GraphConfig.SaveGraph;
    //запис у файл
    writeln(GraphConfigFile,GraphConfigStr);
    writeln(GraphConfigFile,GraphConfigExArr[i].ShowPreview);
    writeln(GraphConfigFile,GraphConfigExArr[i].CamFunc);
    writeln(GraphConfigFile,GraphConfigExArr[i].TimerDelay);
```

```
  end;

  //закриття файлу
  CloseFile(GraphConfigFile);
  {$I+}
  if IOResult <> 0 then Result := false;
end;
//---------------------------------i

{ функція ініціалізує камери
VideoDeviceList - список всіх доступних камер
у випадку удачі функція повертає true, інакше false}
function InitCams(var GraphConfigExArr: TGraphConfigExArr;
                        VideoDeviceList: TStringList): boolean;
var
  GraphConfigFile: TextFile; //файлова змінна
  CurLine: String; //рядок файлу
  i: integer; //лічильник
  CamsInitStat: array of boolean; //стан ініціалізованості камер
  TmpConfig: TGraphConfig;//тимчасове зберігання налаштувань

begin
  Result := true;

  //розміри масивів
  SetLength(GraphConfigExArr,VideoDeviceList.Count);
  SetLength(CamsInitStat,VideoDeviceList.Count);

  //заповнюємо
  for i := 0 to High(CamsInitStat) do
    CamsInitStat[i] := false;

  //якщо є файл із налаштуваннями відео - зчитуємо з нього рядка налаштувань
  if FileExists(GraphConfigFileName) then
  begin
    //ініціалізація
    TmpConfig := TGraphConfig.Create;

    //присвоєння файлу
    AssignFile(GraphConfigFile,GraphConfigFileName);

    //проходимо за списком доступних камер і шукаємо їхнього налаштування у
файлі
    for i := 0 to VideoDeviceList.Count - 1 do
    begin
      {$ I-I-}
      Reset(GraphConfigFile);

      //поки не скінчився файл або поки не знайшли налаштування поточної камери
      while (not EOF(GraphConfigFile)) and (not CamsInitStat[i]) do
      begin
        //читаємо рядок з головними налаштуваннями
        readln(GraphConfigFile,CurLine);
        TmpConfig.RestoreGraph(CurLine);

        //якщо ці налаштування для поточної камери, те привласнюємо їх їй
        if TmpConfig.VCapSource = VideoDeviceList.Strings[i] then
        begin
          GraphConfigExArr[i].GraphConfig := TGraphConfig.Create;

          GraphConfigExArr[i].GraphConfig.RestoreGraph(CurLine);

          readln(GraphConfigFile,CurLine);
          GraphConfigExArr[i].ShowPreview := StrToInt(CurLine);

          readln(GraphConfigFile,CurLine);
          GraphConfigExArr[i].CamFunc := StrToInt(CurLine);

          readln(GraphConfigFile,CurLine);
```

```
          GraphConfigExArr[i].TimerDelay := StrToInt(CurLine);

          //ця камера проиниціалазирована!
          CamsInitStat[i] := true;
        end
        else
        begin
          //перелистуємо 3-и рядка з іншими налаштуваннями
          readln(GraphConfigFile);
          readln(GraphConfigFile);
          readln(GraphConfigFile);
        end;
      end;

      //закриття файлу
      CloseFile(GraphConfigFile);
      {$I+}
      if IOResult <> 0 then
      begin
        Result := false;
        Exit;
      end;
    end;
  end;
  //якщо залишилися не проініціалізовані камери, ініціалізуємо їх за
замовчуванням
  for i := 0 to High(CamsInitStat) do
  begin
    if not CamsInitStat[i] then
      DefaultInitCam(VideoDeviceList.Strings[i],GraphConfigExArr[i]);
  end;
end;
//---------------------------------

//процедура виводить в ComboBox доступні відео режими на компоненті VideoCapture
procedure ShowVideoModes(ComboBox: TComboBox; VideoCapture: TVideoCapture);
var
  i: integer; //лічильник
  CurVideoMode: TVCapMode; //поточний відео режим
begin
  ComboBox.Clear; //очищення
  CurVideoMode := VideoCapture.VCapMode; //запам'ятовуємо тек. відео режим

  //у циклі виводимо доступні й визначаємо поточний відео режими
  for i := 0 to VideoCapture.VCapModeCount - 1 do
  begin
    ComboBox.Items.Add(GetModeString(VideoCapture.VCapModes[i]));
    if IsEqualModes(CurVideoMode,VideoCapture.VCapModes[i]) then
      ComboBox.ItemIndex := i;
  end;
end;
//---------------------------------

//процедура ініціалізує відео налаштування для камери CamName за замовчуванням
procedure DefaultInitCam(CamName: String; var GraphConfigEx: TGraphConfigEx);
begin
 //створення об'єкта основних налаштувань камери
 GraphConfigEx.GraphConfig := TGraphConfig.Create;

 with GraphConfigEx.GraphConfig do
  begin
    //ім'я камери
    VCapSource := CamName;

    //аудіо пристрою не використовуємо
    ACapSource := '';

    //аудіо компресори не використовуємо
    AComp := '';
```

```
    //імена файлів для запису відео
    CaptureFileName := 'Capture.avi';
    TempCaptureFileName := 'TempCapture.avi';
    PreallocFileSize := 0;

    //що хочемо одержати
    WantCapture := false;
    WantPreview := false;
    WantBitmaps := false;

    //аудіо не потрібно
    WantAudio := false;
    WantDVAudio := false;
    WantAudioPreview := false;

    //тимчасовий файл
    UseTempFile := true;
    DoPreallocFile := false;
    PreallocFileSize := 0;

    //формат пікселя
    PixelFormat := pfDevice;
    //???
    DVResolution := dvrDontWorry;

    //налаштування відео режиму
    with VCapMode do
    begin
      MediaType := MEDIATYPE_Video;
      MediaSubType := MEDIASUBTYPE_RGB24;
      Width := 640;
      Height := 480;
      BitCount := 24; //???
      FrameRate := 30.000;
      MinFrameRate := 30.000;
      MaxFrameRate := 30.000;
    end;
  end;
  GraphConfigEx.ShowPreview := 0; //потрібно чи показувати форму із зображенням
  GraphConfigEx.CamFunc := CamFuncNone; //камера не функціональна
  GraphConfigEx.TimerDelay := 65; //частота обробки 65 мс
end;
//---------------------------------

{допоміжні процедури, що входять у клас форми
 у всіх процедурах: CamNum - номер камери,
 над якою виробляється дія}

{Для розгорнення}
//настроїти параметри циклічного розгорнення
procedure TMainForm.ConfigRazvertka(CamNum: Cardinal);
begin
  //набудовуємо перше розгорнення
  URazvertkaConfig.LocalRazvertkaConfig := @RazvertkaArr[CamsListBox.ItemIndex];
  RazvertkaConfigForm.ShowModal;
  RazvertkaArr[CamsListBox.ItemIndex].SetRazvertkaConfig(LocalRazvertkaConfig^);
end;
//---------------------------------

{Оброблювачі подій}

//Подія - перед створенням форми
{{Тут відбувається ініціалізація налаштувань компонентів і змінних}
procedure TMainForm.FormCreate(Sender: TObject);
var
  VideoDeviceList: TStringList; //список відео пристроїв
  Res1,Res2: HRESULT;
begin
```

```
  {для text to speech}
  {Ініціалізація аудіопристрою}
  Res1 := CoCreateInstance(CLSID_MMAudioDest, Nil,
CLSCTX_ALL,IID_IAudioMultiMediaDevice, fIAMM);
  {Створення об'єкта, що перераховується, для перебору всіх движків у системі за
допомогою інтерфейсу ITTSEnum}
  Res2 := CoCreateInstance(CLSID_TTSEnumerator, Nil,
CLSCTX_ALL,IID_ITTSEnum,aTTSEnum);

  if (Res1 = S_OK) and (Res2 = S_OK) then
  begin
    aTTSEnum.Reset;//Скидаємо на перший
    {Одержуємо другий движок}
    aTTSEnum.Next(1, ModeInfo, @NumFound);
    aTTSEnum.Next(1, ModeInfo, @NumFound);
    aTTSEnum.Select(ModeInfo.gModeID, fITTSCentral, IUnknown(fIAMM));
    {Одержуємо інші}
    {While NumFound > 0 do
     begin
       ComboBox1.Items.Add(String(ModeInfo.szModeName));
       aTTSEnum.Next(1, ModeInfo, @NumFound);
     end;}
  end;
  //ініціалізація модуля спілкування
  Communication := TCommunication.Create;
  Communication.Start;

  //створюємо об'єкт системи координат
  SystemKoordinat := TSystemKoordinat.Create;
  SystemKoordinat.Start;

  //одержуємо список доступних камер
  VideoDeviceList := GetVideoDevicesList();

  //запуск головного потоку робосервера
  if RoboRootServerActive then
    RoboRootServerManageThread := TRoboRootServerManagThread.Create(false);

  //якщо є хоча б одна камера
  if VideoDeviceList.Count > 0 then
    URoboRootServer.VideoExist := true;

  //потрібно проініціалізувати налаштування камер
  InitCams(GraphConfigExArr,VideoDeviceList);

  //заповнюємо список камер
  CamsListBox.Items.Assign(VideoDeviceList);

end;
//---------------------------------

//Подія - перед відображенням форми
//Тут активуються компоненти й настроюються керуючі елементи
//(кнопки, форми й т.д.)
procedure TMainForm.FormShow(Sender: TObject);
var
  i: integer; //лічильник
begin
  //створення масиву компонентів для роботи з камерами
  SetLength(VideoCaptureArr,Length(GraphConfigExArr));
  for i := 0 to High(VideoCaptureArr) do
  begin
    VideoCaptureArr[i] := TVideoCapture.CreateParented(PreviewWindow.Handle);
    VideoCaptureArr[i].RestoreGraph(GraphConfigExArr[i].GraphConfig);
  end;

  //завдання розміру масиву оброблювачів захоплення кадру
  SetLength(BitmapGrabEventArr,Length(GraphConfigExArr));
```

```
  //створення об'єктів оброблювачів кадрів
  for i := 0 to High(GraphConfigExArr) do
  begin
    BitmapGrabEventArr[i] := TBitmapGrabEvent.Create(i);
    VideoCaptureArr[i].OnBitmapGrabbed := BitmapGrabEventArr[i].AnalyzBitmap;
  end;

  //завдання розміру масиву таймерів камер
  SetLength(CamTimerArr,Length(GraphConfigExArr));

  //створення й запуск таймерів для потрібних камер
  for i := 0 to High(GraphConfigExArr) do
  begin
    CamTimerArr[i] := TCamTimer.Create(i);
    if GraphConfigExArr[i].GraphConfig.WantBitmaps then
    begin
      CamTimerArr[i].StartTimer(GraphConfigExArr[i].TimerDelay);
    end;
  end;

  //задаємо розмір масивам розгорнення
  SetLength(RazvertkaArr,Length(GraphConfigExArr));
  SetLength(RazvertkaConfigArr,Length(GraphConfigExArr));

  //задаємо налаштування для розгорнень
  for i := 0 to High(RazvertkaArr) do
  begin
    with RazvertkaConfigArr[i] do
    begin
      a := VideoCaptureArr[i].VCapMode.Width div 2;
      a_corr := 5;
      b := VideoCaptureArr[i].VCapMode.Height div 2;
      y_offset := 0;
      klaster_size := 5;
      porog := 50;
      step := 3;
      fi_step := 0.005;
      RazvertkaArr[i] := TRazvertka.Creat(RazvertkaConfigArr[i]);
    end;
  end;
  //кнопка для руху
  if not Communication.PortEn then
  begin
   // RisovalkaButton.Enabled := false;
  end;
end;
//---------------------------------


//Подія - перед закриттям форми
procedure TMainForm.FormClose(Sender: TObject; var Action: TCloseAction);
var
  i: integer;
begin
  //зупинка системи координат
  SystemKoordinat.Stop;
  SystemKoordinat.Destroy;


  //зупинка каналу
  Communication.Destroy;

  //зупинка робосервера
    if RoboRootServerActive then
    RoboRootServerManageThread.Terminate;
```

```
  //зупинка таймерів камер
  for i := 0 to High(CamTimerArr) do
  begin
    if CamTimerArr[i] <> nil then
      CamTimerArr[i].Destroy;
  end;
  SetLength(CamTimerArr,0);

  //знищення оброблювачів події захоплення кадру з камер
  for i := 0 to High(BitmapGrabEventArr) do
  begin
    if BitmapGrabEventArr[i] <> nil then
      BitmapGrabEventArr[i].Destroy;
  end;
  SetLength(BitmapGrabEventArr,0);

  //зупинка й знищення об'єктів для захоплення відео
  for i := 0 to High(VideoCaptureArr) do
  begin
    if VideoCaptureArr[i].Capturing then
      VideoCaptureArr[i].StopCapture;
    if VideoCaptureArr[i].Previewing then
      VideoCaptureArr[i].StopPreview;
    VideoCaptureArr[i].Destroy;
  end;
  SetLength(VideoCaptureArr,0);

  //знищення розгорнень і їхніх налаштувань
  for i := 0 to High(RazvertkaArr) do
  begin
    RazvertkaArr[i].Destroy;
  end;
  SetLength(RazvertkaArr,0);
  SetLength(RazvertkaConfigArr,0);
end;
//-----------------------------------

//Подія - натискання на "Руки"
procedure TMainForm.MovingFormButtonClick(Sender: TObject);
begin
  MovingForm.ShowModal;
end;
//-----------------------------------

//Подія - Натискання "Сервер"
procedure TMainForm.Button3Click(Sender: TObject);
begin
  RoboServerConfigForm.ShowModal;
end;
//-----------------------------------

//Подія - Натискання на "Рух по смузі"
procedure TMainForm.MoveForLineClick(Sender: TObject);
begin
  ForwSpeed := 30;
  MoveForLineFlag := not MoveForlineFlag;
end;
//-----------------------------------

//Подія - натискання на "Обновити" (список камер)
procedure TMainForm.RefreshCamsListButtonClick(Sender: TObject);
var
  VideoDeviceList: TStringList; //список камер
  i: integer; //лічильник
begin
  //одержуємо список камер
  VideoDeviceList := GetVideoDevicesList(true);

  //ініціалізуємо камери заново
```

```
  InitCams(GraphConfigExArr,VideoDeviceList);

  //знищення об'єктів для вже неіснуючих камер
  for i := VideoDeviceList.Count to High(VideoCaptureArr) do
  begin
    CamTimerArr[i].StopTimer;
    CamTimerArr[i].Destroy;
    BitmapGrabEventArr[i].Destroy;
  end;

  //установлюємо нові розміри масивів
  SetLength(VideoCaptureArr,VideoDeviceList.Count);
  SetLength(BitmapGrabEventArr,VideoDeviceList.Count);
  SetLength(CamTimerArr,VideoDeviceList.Count);
  //у циклі створюємо потрібні об'єкти
  for i := 0 to High(VideoCaptureArr) do
  begin
    //об'єкти для захоплення відео
    if VideoCaptureArr[i] = nil then
    begin
      VideoCaptureArr[i] := TVideoCapture.CreateParented(PreviewWindow.Handle);
      VideoCaptureArr[i].RestoreGraph(GraphConfigExArr[i].GraphConfig);
    end;
    //події захоплення кадру
    if BitmapGrabEventArr[i] = nil then
    begin
      BitmapGrabEventArr[i] := TBitmapGrabEvent.Create(i);
      VideoCaptureArr[i].OnBitmapGrabbed := BitmapGrabEventArr[i].AnalyzBitmap;
    end;
    //таймери для захоплення кадрів
    if (CamTimerArr[i] = nil) then
    begin
      CamTimerArr[i] := TCamTimer.Create(i);
      if GraphConfigExArr[i].GraphConfig.WantBitmaps then
        CamTimerArr[i].StartTimer(GraphConfigExArr[i].TimerDelay);
    end;
  end;

  //заповнюємо список на формі
  CamsListBox.Clear;
  for i := 0 to VideoDeviceList.Count - 1 do
    CamsListBox.Items.Add(VideoDeviceList.Strings[i]);
end;
//-----------------------------------

//Подія - натискання на "Налаштування" (обраної камери)
procedure TMainForm.CamConfigButtonClick(Sender: TObject);
begin
  if CamsListBox.ItemIndex >= 0 then
  begin
    //передаємо індекс налаштувань обраної камери в модуль налаштування камери
    UGraphConfig.GraphConfigExIndex := CamsListBox.ItemIndex;
    //виводимо форму налаштувань камери в модальному режимі
    if UGraphConfig.GraphConfigForm.ShowModal = mrOK then
    begin
      //зберігаємо й відновлюємо налаштування камери
      SaveGraphConfig(GraphConfigExArr);

VideoCaptureArr[CamsListBox.ItemIndex].RestoreGraph(GraphConfigExArr[CamsListBox
.ItemIndex].GraphConfig);

      //запускаємо або перезапускаємо або зупиняємо таймери якщо потрібно
      if GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.WantBitmaps then
      begin
        if CamTimerArr[CamsListBox.ItemIndex].Active then
          CamTimerArr[CamsListBox.ItemIndex].StopTimer;

CamTimerArr[CamsListBox.ItemIndex].StartTimer(GraphConfigExArr[CamsListBox.ItemI
ndex].TimerDelay);
```

```
      end
      else
      begin
        if CamTimerArr[CamsListBox.ItemIndex].Active then
          CamTimerArr[CamsListBox.ItemIndex].StopTimer;
      end;

      //якщо потрібно показувати зображення на екрані
      if GraphConfigExArr[CamsListBox.ItemIndex].ShowPreview = 1 then
      begin
        ActiveCamIndex := CamsListBox.ItemIndex;
        PreviewWindow.ShowEx;
      end
      else
      begin
        PreviewWindow.Hide;
        ActiveCamIndex := -1;
      end;
      //обновляємо діалоги й відео режими

ShowAvialableDialogs(DialogListBox,VideoCaptureArr[CamsListBox.ItemIndex]);
      ShowVideoModes(VideoModeComboBox, VideoCaptureArr[CamsListBox.ItemIndex]);

      //кнопки
      if GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.WantBitmaps then
      begin
        AnalyzFrameButton.Enabled := true;
        OnFlyDebugButton.Enabled := true;
      end
      else
      begin
        AnalyzFrameButton.Enabled := false;
        OnFlyDebugButton.Enabled := false;
      end;
    end;
  end
  else
    ShowMessage('Не обрана камера!');
end;
//-----------------------------------

//подія - натискання на Списку камер
procedure TMainForm.CamsListBoxClick(Sender: TObject);
begin
  //виводимо картинку якщо потрібно
  if GraphConfigExArr[CamsListBox.ItemIndex].ShowPreview = 1 then
  begin
    ActiveCamIndex := CamsListBox.ItemIndex;
    PreviewWindow.ShowEx;
  end
  else
  begin
    PreviewWindow.Hide;
    ActiveCamIndex := -1;
  end;
  //виводимо діалоги й відео режими для обраної камери
  ShowAvialableDialogs(DialogListBox,VideoCaptureArr[CamsListBox.ItemIndex]);
  ShowVideoModes(VideoModeComboBox, VideoCaptureArr[CamsListBox.ItemIndex]);

  //ім'я камери
  PreviewWindow.Caption :=
GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.VCapSource;

  //кнопки
  if GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.WantBitmaps then
  begin
    AnalyzFrameButton.Enabled := true;
    OnFlyDebugButton.Enabled := true;
  end
```

```
  else
  begin
    AnalyzFrameButton.Enabled := false;
    OnFlyDebugButton.Enabled := false;
  end;
end;
//-----------------------------------

//подія - подвійний натискання на списку діалогів камери
procedure TMainForm.DialogListBoxDblClick(Sender: TObject);
var
  i: integer; //лічильник
begin
  //шукаємо виділений рядок, щоб викликати діалог, ім'я якого в ній написано
  for i := 0 to DialogListBox.Count - 1 do
    if DialogListBox.Selected[i] then
    begin
      MainForm.Enabled := false;

UPreview.VideoCaptureArr[CamsListBox.ItemIndex].ShowDialog(TCaptureDialog(Dialog
ListBox.Items.Objects[i]));
      MainForm.Enabled := true;
    end;
  //зберігаємо новий відео режим
  GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.VCapMode :=
VideoCaptureArr[CamsListBox.ItemIndex].VCapMode;
  SaveGraphConfig(GraphConfigExArr);
  //виводимо сталий режим у списку відео режимів

ShowVideoModes(VideoModeComboBox,UPreview.VideoCaptureArr[CamsListBox.ItemIndex]
);
end;
//-----------------------------------

//Подія - зміна в списку відео режимів
procedure TMainForm.VideoModeComboBoxChange(Sender: TObject);
begin
  //Установлюємо новий відео режим

VideoCaptureArr[CamsListBox.ItemIndex].SetVCapMode(VideoModeComboBox.ItemIndex);
  //змінюємо налаштування
  GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.VCapMode :=
    VideoCaptureArr[CamsListBox.ItemIndex].VCapMode;
  //зберігаємо налаштування
  SaveGraphConfig(GraphConfigExArr);
  //якщо потрібно, те возобнавляем висновок на екран
  if GraphConfigExArr[CamsListBox.ItemIndex].ShowPreview = 1 then
    VideoCaptureArr[CamsListBox.ItemIndex].StartPreview;
end;
//-----------------------------------

//Подія - натискання на "Маяки"
procedure TMainForm.CommunicationButtonClick(Sender: TObject);
begin
  CommunicationForm.Show;
end;
//-----------------------------------

//Подія - натискання на "Рисовалка"
procedure TMainForm.RisovalkaButtonClick(Sender: TObject);
begin
 OKRightDlg.Visible:=true;
// MainForm.Hide;
//  URisovalka.RisovalkaForm.Visible := true;
end;
//------------------+-----------------i

//Подія - Натискання на "Аналіз Кадру"
procedure TMainForm.AnalyzFrameButtonClick(Sender: TObject);
```

```
begin
  if CamsListBox.ItemIndex >= 0 then
  begin
    BitmapGrabEventArr[CamsListBox.ItemIndex].AnalyzFrame := true;
    FrameForm.ShowModal;
  end
  else
    ShowMessage('Не обрана камера!');
end;
//-----------------------------------

//Подія - натискання на "Розгорнення"
procedure TMainForm.RazvertkaButtonClick(Sender: TObject);
begin
  //виводимо форму налаштувань розгорнення
  if CamsListBox.ItemIndex >= 0 then
  begin
    URazvertkaConfig.LocalRazvertkaConfig :=
@RazvertkaConfigArr[CamsListBox.ItemIndex];
    RazvertkaConfigForm.ShowModal;

RazvertkaArr[CamsListBox.ItemIndex].SetRazvertkaConfig(URazvertkaConfig.LocalRaz
vertkaConfig^);
  end
  else
    ShowMessage('Не обрана камера!');
end;
//-----------------------------------

//Подія - натискання на "дебаг на льоту"
procedure TMainForm.OnFlyDebugButtonClick(Sender: TObject);
begin
  //якщо обрано камеру
  if CamsListBox.ItemIndex >= 0 then
  begin
    //установлюємо прапорець для дебага на льоту й виводимо форму
    BitmapGrabEventArr[CamsListBox.ItemIndex].OnFlyDebug := true;
    FrameForm.ShowModal;
  end
  else
    ShowMessage('Не обрана камера!');
end;
//-----------------------------------


procedure TMainForm.Button1Click(Sender: TObject);
begin
  Communication.stopsignal := 1;
end;

procedure TMainForm.Button2Click(Sender: TObject);
begin
  Communication.stopsignal := 0;
end;
//-----------------------------------

//Подія - натискання "Семпли"
procedure TMainForm.AddSampleButtonClick(Sender: TObject);
begin
  if CamsListBox.ItemIndex >= 0 then
  begin
    BitmapGrabEventArr[CamsListBox.ItemIndex].NeedAddSample := true;
  end
  else
    Picture_processing_Hamming.AddSampleForm.ShowModal;
end;

procedure TMainForm.RestaranButtonClick(Sender: TObject);
begin
```

```
  Restaran := TRestaran.Create;
  Restaran.Start(1);
end;

procedure TMainForm.Button4Click(Sender: TObject);
begin
  UDebug.DebugForm.Show;
end;

procedure TMainForm.Button6Click(Sender: TObject);
begin
 //mbDisconnect();
 //MainForm.Label3.Caption:='Порт закритий';
end;



function Send:integer;
begin
   result:=mbWriteHoldingRegisters(1,@(Communication.RecvBuff),0,3);
 // mbWriteHoldingRegisters(1,@SendBuff,0,4);
 //result:=SENDMSG(SendMsNb, CANSendBuff)
end;


procedure TMainForm.Button5Click(Sender: TObject);
var
  len_written: integer;
  speeds: array [0..1] of smallint;
begin
  //mbSetLogDetails(true,true,true);
  mbSetLogDetails(false,false,false);
  i:=mbConnect(port,115200,1,0,3);

  if i=0 then
  begin
   MainForm.Label3.Caption:='не вдалося відкрити порт';
  end
  else
  begin
   MainForm.Label3.Caption:='порт відкритий';


   mbReset(1);
   sleep(10);

   mbExecuteProgramFile(1,'image.raw');

   sleep(10);
   mbReportDeviceID(1,dest1,250,(@len_written));

   Communication.PortEn:=true;
  end;



   MainForm.Timer1.Enabled:=true;

   //Speeds[0]:=100;
   //Speeds[1]:=-100;

  // Communication.RecvBuff.w1:=100;
  // Communication.RecvBuff.w2:=-100;
  // Communication.RecvBuff.w3:=0;

   // Send;
  // mbWriteHoldingRegisters(1,@(Communication.RecvBuff),0,3);
end;
```

```
procedure TMainForm.Timer1Timer(Sender: TObject);
begin
//ghgf
end;

procedure TMainForm.Button7Click(Sender: TObject);
var i1,count,tmp,i:integer;
n_debug:textfile;
begin

    assignfile(n_debug,neuro_debug_file_name);
    append(n_debug);

    count:=0;
    tmp:=AddSampleForm.Cognitive_graphic_BP1.LayerCount-1;
    NeuroCount.Text:='';

    //    for i1:=0 to length(xInputVector) do
    //    write(n_debug,inttostr(round(xInputVector[i1])));
    //    writeln(n_debug,'');

    addsampleform.Cognitive_graphic_BP1.Compute(xInputVector);


    //addsampleform.Cognitive_graphic_BP1.
    for i := 0 to length(xOutputVector)-1 do
    begin

xOutputVector[i]:=AddSampleForm.Cognitive_graphic_BP1.LayersBP[tmp].Neurons[i].O
utput;
    NeuroCount.Text:=NeuroCount.Text+'-'+floattostr(xOutputVector[i]);
    end;
    //if AddSampleForm.Layers[1].Neurons[i].Output = 1 then
      //Count:=count+1;

  // Нейрона мережа Хопфілда
    {addsampleform.Cognitive_graphic_Hopf1.Calc;

  for i := 0 to IconSize* IconSize-1 do
    if AddSampleForm.Cognitive_graphic_Hopf1.Layers[1].Neurons[i].Output = 1
then
      Count:=count+1;

  NeuroCount.Text:=inttostr(count);
 }
    closefile(n_debug)
end;

procedure TMainForm.Button8Click(Sender: TObject);
begin
Form5.Show;
end;

end.
```

Файл UPreview.pas – модуль роботи з камерами

```
unit UPreview;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, VCap,USingleFrame,UTypeConst,URazvertka, UCamFunc,URoboRootServer,
  ExtCtrls,UMoving,Picture_Analyz_Hamming,UQPixels,Picture_processing_Hamming;

type
  //захоплений кадр
  TCapturedBitmap = Vcap.TCapturedBitmap;

  TBitmap = Graphics.TBitmap;

  //клас який описує подію захоплення кадру з камери
  TBitmapGrabEvent = class
    private
      EventIndex: Integer; //індекс події (номер камери)
    public
      AnalyzFrame: boolean; //аналіз кадру
      OnFlyDebug: boolean; //дебаг на льоту
      NeedAddSample: boolean; //потрібно додати семпл
      constructor Create(Index: Cardinal); //конструктор
      destructor Destroy; override; //деструктор
      //аналіз захопленого кадру
      procedure AnalyzBitmap(Bitmap: TCapturedBitmap);
  end;

  //тип масиву оброблювачів захоплення кадру з камери
  TBitmapGrabEventArr = array of TBitmapGrabEvent;


  TPreviewWindow = class(TForm)
    Video: TImage;
    procedure VideoCaptureDeviceLost(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure ShowEx;
  end;

var
  PreviewWindow: TPreviewWindow; //форма для висновку зображення з камери
  VideoCaptureArr: TVideoCaptureArr; //масив компонентів для захоплення відео
  BitmapGrabEventArr: TBitmapGrabEventArr;  //масив оброблювачів захоплення
кадру з камери
  QP: TQuickPixels;

  UgliPolosi: TUgliRazrivov; //кути можливих напрямків руху
  Flag: boolean = false;

  implementation

uses
  Main_Cognitive_graphic_,UCamTimers,Urestaran;

var
  F: TextFile;
//  BitmapArr: TByteArr;

{$R *.dfm}

//конструктор
```

```
constructor TBitmapGrabEvent.Create(Index: Cardinal);
begin
  inherited Create;
  //індекс оброблювача
  EventIndex := Index;
  //аналіз кадру
  AnalyzFrame := false;
  //дебаг на льоту
  OnFlyDebug := false;
end;
//---------------------------------

//деструктор
destructor TBitmapGrabEvent.Destroy;
begin
  inherited Destroy;
end;
//---------------------------------

//аналіз захопленого бітмапа
procedure TBitmapGrabEvent.AnalyzBitmap(Bitmap: TCapturedBitmap);
var
  RazvertkaTlumitsya: TRazvertka;
  RazvertkaTlumitsyaConfig: TRazvertkaConfig;
  razriv_cnt,razriv_cnt_tlumitsya: integer; // кількість розривів
  UgliRazrivov,UgliRazrivovTlumitsya: TUgliRazrivov; //кути розривів
//  BitmapArr: TByteArr;
  y: Cardinal;
begin
  //якщо потрібно давати відео робосерверу
  if URoboRootServer.VideoInUse and (ActiveCamNum = EventIndex) then
  begin
    if not KadrSending then
    begin
      KadrFormating := true;
      Kadr.Assign(Bitmap);
      //KadrSizeWH := Bitmap.Width;
      KadrFormating := false;
    end;
  end;

  //аналіз одного кадру з виводом інформації про кластери
  if AnalyzFrame and not OnFlyDebug then
  begin
    USingleFrame.FrameForm.DebugFrame(Bitmap,true);
    AnalyzFrame := false;
  end;

  //аналіз потоку кадрів без висновку інформації про кластери
  if OnFlyDebug and not AnalyzFrame then
  begin
    USingleFrame.FrameForm.DebugFrame(Bitmap);
  end;

  //якщо потрібно взяти семпл
  if NeedAddSample then
  begin
    Picture_processing_Hamming.AddSampleForm.SampleImage.Picture.Assign(Bitmap);
    NeedAddSample := false;
    with AddSampleForm do
    begin
      SampleImage.Width := Bitmap.Width;
      SampleImage.Height := Bitmap.Height;
      {
      RadioTSample.Top := SampleImage.Height;
      RadioNotTSample.Top := SampleImage.Height;
      AddSampleButton.Top := SampleImage.Height + RadioTSample.Height;
      BrowseButton.Top := SampleImage.Height + RadioTSample.Height;
      }
```

```
      //height:=GroupBox1.Height+SampleImage.Height;
      //ObjectImage.Left:=SampleImage.Width;
      ObjectImage.Left:=0;
      GroupBox1.top:=SampleImage.Height;  //AddSampleForm.height
      //AddSampleForm.ClientWidth := SampleImage.Width+;
      if not(visible) then ShowModal;
    end;
  end;

  //якщо обрано камеру подія якого відбулося
  if (EventIndex = ActiveCamIndex) then
  begin
    PreviewWindow.ClientWidth := Bitmap.Width;
    PreviewWindow.ClientHeight := Bitmap.Height;
    PreviewWindow.Video.Picture.Bitmap.Assign(Bitmap);

   // Main_Cognitive_graphic_.MainForm.TLabel.Caption := 'немає!';
  end;

  //рух по полігону, обробляємо обрану камеру
  if GraphConfigExArr[EventIndex].CamFunc = CamFuncPolygonForw then
  begin
    //розгорнення
    razriv_cnt := RazvertkaArr[EventIndex].Klaster_Analiz(UgliRazrivov,Bitmap);
    //смуга
    Polosa(razriv_cnt,UgliRazrivov,UgliPolosi);
    //може бути T
    //if razriv_cnt = 2 then


    if recognition_run then
    begin
      QP.Attach(Bitmap);

      SetLength(BitmapArr,QP.Height);
      for y := 0 to High(BitmapArr) do
        SetLength(BitmapArr[y],QP.Width);
      ConvertBitmapToMonoChrome(QP,BitmapArr);
      //segment(BitmapArr);

      //TSampleRule(BitmapArr);

        RobotRecognition.BitmapReady:=true;
    end;
  end;

  //тлумиться
  if Restaran <> nil then
  if Restaran.tlumitsya = 1 then
  begin
    with RazvertkaTlumitsyaConfig do
    begin
      a := RazvertkaConfigArr[EventIndex].a;
      a_corr := RazvertkaConfigArr[EventIndex].a_corr;
      b := 10;
      y_offset := RazvertkaConfigArr[EventIndex].y_offset;
      klaster_size := RazvertkaConfigArr[EventIndex].klaster_size;
      porog := RazvertkaConfigArr[EventIndex].porog;
      step := RazvertkaConfigArr[EventIndex].step;
      fi_step := RazvertkaConfigArr[EventIndex].fi_step;
    end;

    RazvertkaTlumitsya := TRazvertka.Creat(RazvertkaTlumitsyaConfig);
    razriv_cnt_tlumitsya :=
RazvertkaTlumitsya.Klaster_Analiz(UgliRazrivovTlumitsya,Bitmap);

    if razriv_cnt_tlumitsya = 2 then
      Restaran.ugol_tlumitsya := (UgliRazrivov[0] + UgliRazrivov[1]) / 2;
```

```
    RazvertkaTlumitsya.Destroy;
  end;

  //простий рух по смузі
  if MoveForLineFlag then
  begin
    razriv_cnt := RazvertkaArr[EventIndex].Klaster_Analiz(UgliRazrivov,Bitmap);
    Polosa(razriv_cnt,UgliRazrivov,UgliPolosi);
    MoveForLine(UgliPolosi);
  end;
end;
//-----------------------------------

//Показати форму
procedure TPreviewWindow.ShowEx;
begin
  if GraphConfigExArr[ActiveCamIndex].GraphConfig.VCapMode.Width <= 640 then
    PreviewWindow.ClientWidth :=
GraphConfigExArr[ActiveCamIndex].GraphConfig.VCapMode.Width
  else
    PreviewWindow.ClientWidth := 640;
  if GraphConfigExArr[ActiveCamIndex].GraphConfig.VCapMode.Height <= 480 then
    PreviewWindow.ClientHeight :=
GraphConfigExArr[ActiveCamIndex].GraphConfig.VCapMode.Height
  else
    PreviewWindow.ClientHeight := 480;

  Left := Screen.Width - Width;
  Top := 0;

  Show;
end;
//-----------------------------------

//Подія - зв'язок з камерою загублена
procedure TPreviewWindow.VideoCaptureDeviceLost(Sender: TObject);
begin
  ShowMessage('Зв'язок з відео пристроєм загублена!');
end;

initialization
  //QP2 := TQuickPixels.Create;
  QP := TQuickPixels.Create;
  AssignFile(F,'Preview.txt');
  Rewrite(F);

finalization
  QP.Destroy;
  CloseFile(F);

end.
```

Файл UCamTimers.pas – модуль встановлення таймерів камер

```
unit UCamTimers;

interface

uses
  Windows,URazvertka, UPreview, UTypeConst, SysUtils;

type
   //тип процедури спрацьовування таймера як метод класу
   //TTimeProc = procedure(uID, uMsg: UINT; dwUser, dw1, dw2: DWORD) of object;

  //клас таймера камери
  TCamTimer = class
    private
      uidtimer: UINT; //ідентифікатор таймера
      TimerDelay: Cardinal; //період спрацьовування таймеа (мс)
      TimerIndex: Cardinal; //індекс таймера
    public
      Active: boolean; // чиактивний таймер
      Constructor Create(Index: Cardinal);
      Destructor Destroy(); override;
      procedure StartTimer(Delay: Cardinal);
      procedure StopTimer();
  end;

  //масив таймерів камер
  TCamTimerArr = array of TCamTimer;

var
  CamTimerArr: TCamTimerArr;//масив таймерів камер

//оброблювач таймерів
procedure TimeProc(uID, uMsg: UINT; dwUser, dw1, dw2: DWORD); stdcall;

implementation

Constructor TCamTimer.Create(Index: Cardinal);
begin
  inherited Create;
  TimerIndex := Index;
  Active := false;
end;
Destructor TCamTimer.Destroy();
begin
  StopTimer();
  inherited Destroy();
end;
procedure TCamTimer.StartTimer(Delay: Cardinal);
begin
  TimerDelay := Delay;
  uidtimer := timeSetEvent(TimerDelay,1,@TimeProc,TimerIndex,1);
  Active := true;
end;
procedure TCamTimer.StopTimer();
begin
  timeKillEvent(uidtimer);
  Active := false;
end;
//оброблювач таймерів камер
procedure TimeProc(uID, uMsg: UINT; dwUser, dw1, dw2: DWORD); stdcall;
begin
  VideoCaptureArr[dwuser].CaptureFrame;
end;

end.
```

Файл UGraphConfig.pas – модуль налаштування камер

```
unit UGraphConfig;

{Модуль налаштування камери}

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, VCap, Buttons, ExtCtrls, ComCtrls, UTypeConst;


const
  MMInit_WrongDeviceNum = -1; //неправильний номер пристрою
  MMInit_WrongCompNum = -1; //неправильний номер компресора


type
  TGraphConfigForm = class(TForm)
    Label3: TLabel;
    VideoCompBox: TListBox;
    OKBitBtn: TBitBtn;
    GraphConfigGroupBox: TGroupBox;
    WantPreviewCheckBox: TCheckBox;
    PixelFormatComboBox: TComboBox;
    Label4: TLabel;
    CaptureFileNameEdit: TLabeledEdit;
    WantCaptureCheckBox: TCheckBox;
    CamFuncComboBox: TComboBox;
    Label1: TLabel;
    WantBitmapsCheckBox: TCheckBox;
    TimerDelayEdit: TLabeledEdit;
    procedure OKBitBtnClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure WantBitmapsCheckBoxClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  GraphConfigForm: TGraphConfigForm; //форма

  //індекс елемента масиву налаштувань
  GraphConfigExIndex: Cardinal;

implementation

uses
  Main_Cognitive_graphic_;

{$R *.dfm}

//---------------------------------i

{при натисканні кнопки ОК відбувається зчитування налаштувань,
заданих користувачем, і присвоєння їх  переданої змінної}
procedure TGraphConfigForm.OKBitBtnClick(Sender: TObject);
var
  i: integer; //лічильник
  SelVCompNum: integer; //номер обраного відео компресора
begin
  //заповнюємо змінну стану камери
  with GraphConfigExArr[GraphConfigExIndex].GraphConfig do
  begin
```

```
    //заповнюємо поле "відео компресор" ім'ям обраного відео компресора
    SelVCompNum := MMInit_WrongCompNum;
    for i := 0 to VideoCompBox.Count - 1 do
      if VideoCompBox.Selected[i] then
        SelVCompNum := i;

    //якщо не один компресор не обраний
    if SelVCompNum = MMInit_WrongCompNum then
      VComp := ''
    else
      VComp := VideoCompBox.Items[SelVCompNum];

    //потрібно записувати відео?
    if WantCaptureCheckBox.Checked then
      WantCapture := true
    else
      WantCapture := false;

    //за замовчуванням не потрібно зображення
    WantPreview := false;
    WantBitmaps := false;

    //потрібно показувати зображення
    if WantPreviewCheckBox.Checked then
    begin
      GraphConfigExArr[GraphConfigExIndex].ShowPreview := 1;
      WantPreview := true;
    end
    else
    begin
      GraphConfigExArr[GraphConfigExIndex].ShowPreview := 0;
    end;

    //потрібні кадри
    if WantBitmapsCheckBox.Checked then
    begin
     if WantPreview = false then
       WantPreview := true;
     WantBitmaps := true;
    end
    else
      WantBitmaps := false;

    //ім'я файлу в який записується відео
    CaptureFileName := CaptureFileNameEdit.Text;

    // функція камери
    GraphConfigExArr[GraphConfigExIndex].CamFunc := CamFuncComboBox.ItemIndex;

    //частота таймера
    GraphConfigExArr[GraphConfigExIndex].TimerDelay :=
StrToInt(TimerDelayEdit.Text);
  end;

  //вибираємо формат подання пікселя
  with PixelFormatComboBox do
  begin
    if Text = 'Визначається пристроєм' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pfDevice
    else if Text = '1 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf1bit
    else if Text = '4 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf4bit
    else if Text = '8 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf8bit
    else if Text = '15 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf15bit
    else if Text = '16 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf16bit
```

```
    else if Text = '24 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf24bit
    else if Text = '32 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf32bit
    else if Text = 'Налаштовуваний' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pfCustom
    else GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat :=
pfDevice;
  end;
end;
//--------------------------------i

{Подія - перед появою форми
виводимо списки доступних
компресорів для стиску відео й відновлюємо галочки у відповідності
с переданими відео налаштуваннями}
procedure TGraphConfigForm.FormShow(Sender: TObject);
var
  i: integer; //лічильник
  VideoCompList: TStringList; //аркуш доступних відео компресорів
begin
  Caption := 'Налаштування відео для ' +
GraphConfigExArr[GraphConfigExIndex].GraphConfig.VCapSource;

  //одержуємо список відео кодеків
  VideoCompList := GetVideoCompressorsList(true);

  //виводимо в компоненти отриману інформацію
  VideoCompBox.Items.Assign(VideoCompList);

  //жодна рядок не виділений
  VideoCompBox.ItemIndex := -1;

  //відновлюємо номер відео компресора
  for i := 0 to VideoCompBox.Count - 1 do
  begin
    if VideoCompBox.Items[i] =
GraphConfigExArr[GraphConfigExIndex].GraphConfig.VComp then
      VideoCompBox.ItemIndex := i;
  end;

  //відновлюємо галочки "Потрібно зображення"
  if GraphConfigExArr[GraphConfigExIndex].ShowPreview = 1 then
    WantPreviewCheckBox.Checked := true;
  if GraphConfigExArr[GraphConfigExIndex].ShowPreview = 0 then
    WantPreviewCheckBox.Checked := false;

  //відновлюємо галочку "Потрібно записувати відео"
  if GraphConfigExArr[GraphConfigExIndex].GraphConfig.WantCapture then
    WantCaptureCheckBox.Checked := true
  else
    WantCaptureCheckBox.Checked := false;

  //відновлюємо поле "потрібні кадри"
  if GraphConfigExArr[GraphConfigExIndex].GraphConfig.WantBitmaps then
    WantBitmapsCheckBox.Checked := true
  else
    WantBitmapsCheckBox.Checked := false;

  //відновлюємо формат пікселя
  case GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat of
    pfDevice : PixelFormatComboBox.Text := 'Визначається пристроєм';
    pf1bit : PixelFormatComboBox.Text := '1 біт';
    pf4bit : PixelFormatComboBox.Text := '4 біт';
    pf8bit : PixelFormatComboBox.Text := '8 біт';
    pf15bit : PixelFormatComboBox.Text := '15 біт';
    pf16bit : PixelFormatComboBox.Text := '16 біт';
    pf24bit : PixelFormatComboBox.Text := '24 біт';
    pf32bit : PixelFormatComboBox.Text := '32 біт';
```

```
    pfCustom : PixelFormatComboBox.Text := 'Налаштовуваний';
  end;

  //відновлюємо ім'я файлу для запису відео  CaptureFileNameEdit.Text :=
GraphConfigExArr[GraphConfigExIndex].GraphConfig.CaptureFileName;


  // фукції камери
  CamFuncComboBox.ItemIndex := GraphConfigExArr[GraphConfigExIndex].CamFunc;

  //частота таймера
  TimerDelayEdit.Text :=
IntToStr(GraphConfigExArr[GraphConfigExIndex].TimerDelay);

end;

//подія - натискання на "потрібні кадри"
procedure TGraphConfigForm.WantBitmapsCheckBoxClick(Sender: TObject);
begin
  if WantBitmapsCheckBox.Checked then
  begin
    WantPreviewCheckBox.Enabled := true;
  end
  else
  begin
    WantPreviewCheckBox.Checked := false;
    WantPreviewCheckBox.Enabled := false;
  end;
end;

end.
```

Файл Picture_processing_Hamming.pas – обробка розпізнаного образу

```
unit Picture_processing_Hamming;
// один з модулів розпізнавання образів, містить:
// -підготовку образа до розпізнання, з обчисленням інваріантів і поворотом
// -читання, запис та інші операції з Базою даних образів
// -навчання нейромережі
// -збереження й запис вагових коефіцієнтів нейромережі
// -автоматичний підбор коефіцієнтів нейромережі
// -захоплення й додавання нового образа в БД


// навчання нейромережі може проиходить досить довгий час
// змінювана цифра - це середньоквадратична помилка:
// на практиці,  якщо вона скакає кілька мінут в однієї й тої ж величини
// значить пійманий локальний мінімум

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls,Picture_Analyz_Hamming,UQPixels, Grids,
ValEdit,math,
  NeuralBaseComp, NeuralBaseTypes, Spin, ComCtrls;

type
  TAddSampleForm = class(TForm)
    SampleImage: TImage;
    SampleOpenDialog: TOpenDialog;
    SampleSaveDialog: TSaveDialog;
    GroupBox1: TGroupBox;
    Label2: TLabel;
    Label3: TLabel;
    matrix_size: TEdit;
    matrix_size: TEdit;
    NumIcons: TEdit;
    Button1: TButton;
    IconTypeSet: TComboBox;
    Label1: TLabel;
    AddSampleButton: TButton;
    BrowseButton: TButton;
    SaveSampleButton: TButton;
    ObjectImage: TImage;
    cellImage: TImage;
    Button2: TButton;
    Label4: TLabel;
    Label5: TLabel;
    shir: TLabel;
    hjkhjk: TLabel;
    Celx: TLabel;
    sdf: TLabel;
    vis: TLabel;
    Label6: TLabel;
    cely: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    ComboBox1: TComboBox;

    Cognitive_graphic_Hopf1    : TCognitive_graphic_Hopf;
    Cognitive_graphic_BP1: TCognitive_graphic_BP;
    prbEpoch: TProgressBar;
    speEpochCount: TSpinEdit;
    Label9: TLabel;
    sttError: TLabel;
    Button3: TButton;
    neroIMp: TEdit;
    neroAlfa: TEdit;
```

```
    neroRate: TEdit;
    Label10: TLabel;
    Button4: TButton;
    Cognitive_graphic_Extended1: TCognitive_graphic_Extended;
    Button5: TButton;
    Button6: TButton;
    Button7: TButton;
    Button8: TButton;
    Button9: TButton;
    reader: TEdit;
    ComboBox2: TComboBox;
    recType: TComboBox;
    TrackBar1: TTrackBar;
    param: TLabel;
    Timer1: TTimer;
    lFPS: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    procedure BrowseButtonClick(Sender: TObject);
    procedure AddSampleButtonClick(Sender: TObject);
    procedure SaveSampleButtonClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button2Click(Sender: TObject);
    procedure Cognitive_graphic_BP1EpochPassed(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Cognitive_graphic_Extended1EpochPassed(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button9Click(Sender: TObject);
    procedure TrackBar1Change(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
      procedure imFeatures(BitmapArr: TByteArr; pixeltype:integer);

var
  AddSampleForm: TAddSampleForm;
  sample_db,n_debug:textfile;
  sample_db_file_name:string ='data\config.db';
  neuro_debug_file_name:string ='data\debug.txt';
  neuro_teach_log_file_name:string ='data\log.txt';
  neuro_weight_file_name:string ='data\neuro.db';
  BitmapArr3:TByteArr;
  recognition_run:boolean=false;
  xVector: TVectorInt;
  xOutputVector: TVectorFloat;
  xInputVector: TVectorFloat;   // Вхідний вектор
  segment_X,segment_Y:integer;
  char_cnt,x_,y_:cardinal;
  debug:boolean =false;
  tm_tick:integer=0;

  implementation

uses upreview,Main_Cognitive_graphic_;
{$R *.dfm}


// щось для роботи з картинками без камери
```

```
procedure TAddSampleForm.BrowseButtonClick(Sender: TObject);
var
  Bitmap: Tbitmap;
begin
  if SampleOpenDialog.Execute then
  begin
    Bitmap := Tbitmap.Create;
    Bitmap.LoadFromFile(SampleOpenDialog.FileName);
    SampleImage.Width := Bitmap.Width;
    SampleImage.Height := Bitmap.Height;
  //  RadioTSample.Top := SampleImage.Height;
  //  RadioNotTSample.Top := SampleImage.Height;
    //AddSampleButton.Top := SampleImage.Height ;//+ RadioTSample.Height;
    BrowseButton.Top := SampleImage.Height;// + RadioTSample.Height;
    SampleImage.Picture.Assign(Bitmap);
    Bitmap.Destroy;
  end;
end;



// функція обробки бінарного масиву
// 1. обчислення центра мас об'єкта
// 2. обчислення орієнтації об'єкта
// 3. поворот об'єкта
// 4. вихоплювання об'єкта
// 5. вжимання об'єкта до іконки
// багато частин можна прискорити.
procedure imFeatures(BitmapArr: TByteArr;pixeltype:integer);
var x,y,i,j,xc,yc:integer;
    N,sumx,sumy:cardinal;
    buff:integer;
    max_x,max_y:integer;
    canvas: tcanvas;
    tmp,O,SumUx,SumUy,SumUxy,Ux,Uy,Uxy,C:real;
    r:single;
    s,ss:extended;
    BitmapArr4,BitmapArr2: TByteArr;
    al:real;
    x_new,y_new:integer;
    actual_min_x,actual_min_y,actual_max_x,actual_max_y:integer;
    Bporog,BPixelsInCell,ix,iy,new_size_x,new_size_y,by,bx:integer;
    dx,dy,celx,cely:real;
    line:string;
    longest:integer;
    offset:cardinal;

const MinBPixelsPercent =10;

begin
  max_x:=Length(BitmapArr[0])-1;max_y:=Length(BitmapArr)-1;
  sumx:=0;sumy:=0;N:=0;



      // малюємо зелененьку рамочку
    if debug then
    begin
      if AddSampleForm.visible then
        begin
        Canvas:=AddSampleForm.SampleImage.Canvas;
        Canvas.Pen.Color := clGreen;
        Canvas.Pen.Width := 1;

        Canvas.MoveTo(Round(left),Round(top));
        Canvas.LineTo(Round(right),Round(top));
        Canvas.LineTo(Round(right),Round(buttom));
        Canvas.LineTo(Round(left),Round(buttom));
        Canvas.LineTo(Round(left),Round(top));
```

```
//        Canvas.MoveTo(Round(x - 10),Round(y - 10));

        end;
    end;


  // Алгоритм обчислення центра мас образа
   for y:=top to buttom do
    for x:=left to right do
      begin
      //BitmapArr2[y,x]:=0;
      buff:=BitmapArr[y,x];
      if buff=pixeltype then buff:=1 else buff:=0;
      sumx:=sumx+( x-left)*buff;
      sumy:=sumy+( y-top)*buff;
      N:=N+buff;
      end;


  // одержали координати центра мас образа
  xc:=left+round(sumx/N);
  yc:=top+round(sumy/N);

  // для сегментації
  segment_X:=xc;
  segment_Y:=yc;

    // малюємо хрестик там, де визначився центр мас
    if debug then
    begin
      if AddSampleForm.visible then
        begin
        Canvas:=AddSampleForm.SampleImage.Canvas;
        x:=xc;
        y:=yc;
        Canvas.Pen.Color := clRed;
        Canvas.Pen.Width := 2;

        Canvas.MoveTo(Round(x - 10),Round(y + 10));
        Canvas.LineTo(Round(x + 10),Round(y - 10));
        Canvas.MoveTo(Round(x - 10),Round(y - 10));
        Canvas.LineTo(Round(x + 10), Round(y + 10));
        end;
    end;


    // обчислення декількох допоміжних величин
    SumUx:=0;SumUy:=0;
    for y:=top to buttom do
      for x:=left to right do
        begin
        buff:=BitmapArr[y,x];
        if buff<>pixeltype then buff:=0 else buff:=1;
        SumUx:=SumUx+buff*sqr( x-xc);
        SumUy:=SumUy+buff*sqr( y-yc);
        SumUxy:=SumUxy+buff*( y-yc)*( x-xc);
      end;
    Ux:=1/12+SumUx*1/N; Uy:=1/12+SumUy*1/N; Uxy:=1/12+SumUxy*1/N; C:=sqrt(sqr(
Ux-Uy)+4*sqr(Uxy));

    //Обчислюємо орієнтацію об'єкта
    if Uy>Ux then
      begin
      O:=180/pi*ArcTan(( Uy-Ux+C)/(2*Uxy));
      //tmp:=( Uy-Ux+C)/(2*Ux*Uy);
      end
      else
      begin
      O:=180/pi*ArcTan((2*Uxy)/( Ux-Uy+C));
```

```
      //tmp:=(2*Uxy)/( Ux-Uy+C);
      end;



      // малюємо напрямок орієнтації об'єкта
  if debug then
    begin
        Canvas.Pen.Color := clRed;
        Canvas.Pen.Width := 5;

        Canvas.MoveTo(xc,yc);
        x:= trunc(( right-xc) * cos(O/180*pi)+xc);
        y:= trunc(( buttom-yc) * sin(O/180*pi)+yc);
        Canvas.LineTo(x,y);


    end;

   // Повертаємо об'єкт щодо точки центра мас, на кут орієнтації
   // попутно визначаємо точне розташування повернутого образа

   actual_min_x:=max_x;
   actual_min_y:=max_y;
   actual_max_x:=0;
   actual_max_y:=0;


    // оскільки невідомо - де в об'єкта що стирчить - споконвічно робимо
    // квадратну матрицю - довгої з діагональ первісної
    if ( right-left)>( buttom-top) then
      begin
      longest:=round(1.5*( right-left));

      end else
      begin
      longest:=round(1.5*( buttom-top));
      end;

      SetLength(BitmapArr2,round(longest)+1);
      for y := 0 to High(BitmapArr2) do
        SetLength(BitmapArr2[y],round(longest)+1);


   O:=-O*pi/180;
   for y := top to buttom do
   begin
     for x := left to right do
     begin
       if BitmapArr[y,x]=pixelType then
         begin
         al:=arctan2((y - yc), (x - xc));
         r := sqrt(sqr(x - xc) + sqr(y - yc));
         //x_new:= trunc(xc + r * cos(al + O));
         //y_new:= trunc(yc + r * sin(al + O));
         x_new:= trunc(r * cos(al + O)+longest/2);
         y_new:= trunc(r * sin(al + O)+longest/2);

         if x_new<actual_min_x then actual_min_x:=x_new;
         if y_new<actual_min_y then actual_min_y:=y_new;
         if x_new>actual_max_x then actual_max_x:=x_new;
         if y_new>actual_max_y then actual_max_y:=y_new;
         if (y_new<longest)and(x_new<longest) and(y_new>0) and (x_new>0) then
BitmapArr2[y_new,x_new]:=1;
         //Canvas.Pixels[x_new,y_new]:= clGreen;
         end;
     end;
   end;
```

```
    // прощай квадратна матриця, довгої з діагональ початкової- вихоплюємо
    // заново повернений образ
    new_size_x:=actual_max_ x-actual_min_x;
    new_size_y:=actual_max_ y-actual_min_y;



    SetLength(BitmapArr4,new_size_y+1);
    for y := 0 to High(BitmapArr4) do
      SetLength(BitmapArr4[y],new_size_x+1);

    if Debug then
    begin
      Canvas:=AddSampleForm.ObjectImage.Canvas;
      AddSampleForm.ObjectImage.Height :=new_size_y;
      AddSampleForm.ObjectImage.Width :=new_size_x;

    end;

   // перетворюємо уже повернений масив із квадратного в прямокутний по границі
   // потім це можно буде зробити в наступному кроці, поки однаково він
   // щось малює - те можна й залишити тут

    for y := 0 to new_size_y do
      begin
      for x := 0 to new_size_x do
        begin

          ///// !!!!!!!!!!!!!!!!!!!!!!!!

          if ((y+actual_min_y)<(longest)) and ((x+actual_min_x)<(longest)) and
((y+actual_min_y)>0) and ((x+actual_min_x)>0)  then
          BitmapArr4[y,x]:=BitmapArr2[y+actual_min_y,x+actual_min_x];
          if debug then if BitmapArr4[y,x]>0 then Canvas.Pixels[x,y]:=clRed else
Canvas.Pixels[x,y]:= clWhite;
        end;
      end;

    if debug then
    begin
     //AddSampleForm.ObjectImage.Width :=new_size_x;
     //AddSampleForm.ClientWidth := 640;
    end;


    SetLength(BitmapArr3,IConSize+1);
    for y := 0 to High(BitmapArr3) do
      SetLength(BitmapArr3[y],IconSize+1);


    // знаходимо відповідність між осередком іконки масивом
    if new_size_x> IconSize then
      celx:=new_size_x/IconSize else celx:=1;

    if new_size_y> IconSize then
      cely:=new_size_y/IconSize else cely:=1;


      // якась інформація про образ
    if debug then
      begin
       AddSampleForm.shir.Caption:=inttostr(new_size_x);
       AddSampleForm.celx.Caption:=inttostr(round(celx));
       AddSampleForm.vis.Caption:=inttostr(new_size_y);
       AddSampleForm.cely.Caption:=inttostr(round(cely));
      end;

    if debug then
```

```
      begin
      offset:=round((6*IconSize+1)*(char_ cnt-1)); //
      AddSampleForm.cellImage.width:=(6*IconSize+1)*5;//(6*IconSize+1) +offset;
      AddSampleForm.cellImage.height:=(6*IconSize+1);

      end;


    if debug then
      begin
      canvas:=AddSampleForm.cellImage.canvas;
      Canvas.Brush.Color := ClWhite;//Canvas.FillRect(Canvas.ClipRect);
      for x_:=0 to IconSize+offset do

       begin
       Canvas.MoveTo(Round(6*x_),Round(0));
       Canvas.LineTo(Round(6*x_),Round(6*IconSize));
      end;
      for y_:=0 to IconSize do
       begin
       Canvas.MoveTo(round(0),Round(6*y_));
       Canvas.LineTo(Round(6*IconSize+offset),Round(6*y_));
      end;
      end;

      // Алгоритм зменшення зображення  - всі параметри плаваючі

          if debug then    Canvas.Brush.Color := clBlack;

    Bporog := Round(celx * cely / 100 * MinBPixelsPercent);


    dy:=0;
    dx:=0;
    BPixelsInCell := 0;


    for iy := 0 to IconSize - 1 do
      begin

      for ix := 0 to IconSize - 1 do
        begin
         dy:=0;dx:=0;
         while (dy+iy*cely)<((iy+1)*cely) do
          begin
          while (dx+ix*celx)<((ix+1)*celx) do
            begin
                  if (round(dy+iy*cely)<new_size_y) and
(round(dx+ix*celx)<new_size_x) then
                  if BitmapArr4[round(dy+iy*cely),round(dx+ix*celx)] = 1 then
BPixelsInCell := BPixelsInCell + 1;
                  dx:=dx+1;
            end;
            dy:=dy+1;dx:=0;;
          end;

        if BPixelsInCell > BPorog then
          begin  BitmapArr3[iy,ix] := 0; end
          else
          begin  BitmapArr3[iy,ix] := 1; if debug then
canvas.FloodFill(ix*6+3+offset,iy*6+3,canvas.pixels[ix*6+3+offset,iy*6+3],fsSurf
ace);  end;
         BPixelsInCell:=0;


        end;
      end;

    // ну от і все - іконка готова - тепер або розпізнаємо її або
```

```
    // додаємо в БД.
    if debug then
    begin
    for y := 0 to IconSize - 1 do
    begin
      for x := 0 to IconSize - 1 do
         write(F,BitmapArr3[y,x]);
      writeln(F,' ');
    end;
        writeln(F,'--------------');
    end;

end;

// додавання нового семпла - перетворення його в іконку, додавання в масив
procedure TAddSampleForm.AddSampleButtonClick(Sender: TObject);
var
  QP: TQuickPixels;
  BitmapArr: TByteArr;
  Icon: TByteArr;
  y: Cardinal;
  IconType: Cardinal;

begin
  QP := TQuickPixels.Create;
  QP.Attach(SampleImage.Picture.Bitmap);

  SetLength(BitmapArr,QP.Height);
  for y := 0 to High(BitmapArr) do
    SetLength(BitmapArr[y],QP.Width);

    IconType:=IconTypeSet.ItemIndex;
{  if RadioTSample.Checked then
    IconType := Icon
  else
    IconType := IconNot;
 }
  //IconTypes:=IconTypes+1;
  ConvertBitmapToMonoChrome(QP,BitmapArr);
  imFeatures(BitmapArr,1);
  AddSample(BitmapArr,IconType); // реально використовується BitmapArr3

  //Close;
end;

//подія - натискання "Зберегти"
procedure TAddSampleForm.SaveSampleButtonClick(Sender: TObject);
begin
  if SampleSaveDialog.Execute then
  begin
    SampleImage.Picture.Bitmap.SaveToFile(SampleSaveDialog.FileName);
  end;
end;
// читання іконки з файлу
function ReadIcon:TByteArr;
var
  x,y: Integer;
  Arr1:TByteArr;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
  buff:string;
  buff2:char;
  cnt:integer;

begin
  SamePixels := 0;
  Width := IconSize;
  Height := IconSize;
```

```
  SetLength(Arr1,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(Arr1[y],IconSize);

  SetLength(IconArr[High(IconArr)].Icon,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(IconArr[High(IconArr)].Icon[y],IconSize);

  cnt:=0;
  for y := 0 to Height - 1 do
    begin
      for x := 0 to Width - 1 do
      begin
        read(sample_db,buff2);
        Arr1[x,y]:=strtoint(buff2);
            if Arr1[x,y] = 1 then
            begin

            xVector[cnt] := 2;
            xInputVector[cnt] := 0;
            end
            else
            begin
            xVector[cnt] := -1;
            xInputVector[cnt] := 1;
            end;
          cnt:=cnt+1;
        //write(F,Arr1[x,y]);
      end;
      //WRITELN(f);
      readln(sample_db,buff);
    end;

  //Cognitive_graphic_Hopf1.AddPattern(xVector);
  result:=Arr1;

end;


// подія відкриття форми, завантаження семплів з файла, ініціалізація й т.п.
procedure TAddSampleForm.FormShow(Sender: TObject);

  var i1,i,j,jj:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff:string;
    buff_file:textfile;
    max_icon_type:integer;

begin
  AddSampleForm.param.Caption:=inttostr( 100-AddSampleForm.TrackBar1.Position);


  matrix_size.text:=inttostr(IconSize);
  matrix_size.text:=inttostr(IconSize);



  if FileExists(sample_db_file_name) then
    begin
      assignfile(sample_db,sample_db_file_name);
      assignfile(n_debug,neuro_debug_file_name);
      rewrite(n_debug);
      reset(sample_db);

      //IconTypes:=2;
```

```
      readln(sample_db,buff); IconSize:=strtoint(buff);matrix_size.Text:=buff;
      readln(sample_db,buff); IconSize:=strtoint(buff);matrix_size.Text:=buff;
      readln(sample_db,buff); IconTypes:=strtoint(buff);//eIconTypes.Text:=buff;
      readln(sample_db,buff);
SetLength(IconArr,strtoint(buff)+1);NumIcons.Text:=buff;
      // вхідний вектор Нейрона мережа Хопфілда - нейронів стільки ж , скільки й
точок в іконці
      SetLength(xVector, IconSize * IconSize);
      // вихідний вектор багатошарової - дорівнює числу образів
      //IconTypes:=3;
      SetLength(xOutputVector, IconTypes);

      // вхідний вектор багатошарової - дорівнює числу нейронів
      SetLength(xInputVector, IconSize * IconSize);


      //  SetLength(xInputVector, 5);
      //  SetLength(xOutputVector, 1);

      // налаштування нейроної мережі Хопфілда

      AddSampleForm.Cognitive_graphic_Hopf1.LayerCount:=1;
      AddSampleForm.Cognitive_graphic_Hopf1.InputNeuronCount:=IconSize *
IconSize;

      //Cognitive_graphic_Extended1.InputFieldCount:=IconSize * IconSize;
      {

Cognitive_graphic_Extended1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);

Cognitive_graphic_Extended1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);

Cognitive_graphic_Extended1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
      }



      // налаштування багатошарової
      //Cognitive_graphic_BP1.LayerCount:=2;
      //Cognitive_graphic_BP1.LayersBP[0].NeuronCount:=IconSize * IconSize;
      //Cognitive_graphic_BP1.LayersBP[1].NeuronCount:=IconSize * IconSize;
      //Cognitive_graphic_BP1.LayersBP[1].NeuronCount:=IconTypes;
      //Cognitive_graphic_BP1.LayersBP[2].NeuronCount:=IconTypes;

      Cognitive_graphic_BP1.ResetPatterns;




      for i:=0 to High(IconArr) do
        begin
        readln(sample_db,buff); IconArr[i].IconType:=strtoint(buff)+1;


        for j:=1 to IconTypes do
          begin
          if j<>IconArr[i].IconType then xOutputVector[ j-1]:=0 else
xOutputVector[ j-1]:=1;
          end;



        IconArr[i].Icon:=ReadIcon;
        // додаємо вектор навчання нейроної мережі Хопфілда
        AddSampleForm.Cognitive_graphic_Hopf1.AddPattern(xVector);

        // Додаємо вектор багатошарової мережі
        //SetLength(xInputVector, 100);
```

```
        //Cognitive_graphic_Extended1.AddPattern(xInputVector, xOutputVector);
        for i1:=0 to length(xInputVector) do
        write(n_debug,inttostr(round(xInputVector[i1])));
        writeln(n_debug,'');
        // Cognitive_graphic_BP1.AddPattern(xInputVector, xOutputVector);
        //SetLength(xInputVector, 256);
        readln(sample_db,buff);
        end;
        writeln(n_debug,'-----');

        {
        readln(sample_db,buff); IconArr[0].IconType:=strtoint(buff);
        xOutputVector[0]:=0;  xOutputVector[1]:=1;
        IconArr[0].Icon:=ReadIcon; // SetLength(xInputVector, 256);
        //Cognitive_graphic_Extended1.AddPattern(xInputVector, xOutputVector);
        Cognitive_graphic_BP1.AddPattern(xInputVector, xOutputVector);
        readln(sample_db,buff);

        //SetLength(xInputVector, 256);
        readln(sample_db,buff); IconArr[1].IconType:=strtoint(buff);
        xOutputVector[0]:=1;  xOutputVector[1]:=0;
        IconArr[1].Icon:=ReadIcon; //SetLength(xInputVector, 256);
        //Cognitive_graphic_Extended1.AddPattern(xInputVector, xOutputVector);
        Cognitive_graphic_BP1.AddPattern(xInputVector, xOutputVector);
        readln(sample_db,buff);
        }
        // Ініціалізувати ваги Нейроної мережі Хопфілда
        //Cognitive_graphic_Hopf1.InitWeights;



      closefile(n_debug);
      closefile(sample_db);
    end;
  //Cognitive_graphic_BP1.ResetPatterns;

end;


// запис іконки у файл
procedure WriteIcon(Arr1: TByteArr);
var
  x,y: Integer;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
begin
  SamePixels := 0;
  Width := Length(Arr1[Low(Arr1)]);
  Height := Length(Arr1);

  for y := 0 to Height - 1 do
    begin
      for x := 0 to Width - 1 do
      begin
        write(sample_db,Arr1[x,y]);
      end;
    writeln(sample_db,'');
    end;
end;

// закриття форми
procedure TAddSampleForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
  var i,j:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff_file:textfile;
    max_icon_type:integer;
```

```
begin
//  recognition_run:=not(recognition_run);
  if recognition_run then
    begin
    RobotRecognition.Terminate;
    AddSampleForm.Button5.Caption:='ПІШОВ!';
    recognition_run:=not(recognition_run);
    end;

  //IconTypes:=strtoint(eIconTypes.text);

  {
  assignfile(sample_db,sample_db_file_name);
  rewrite(sample_db);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,inttostr(IconTypes));
  writeln(sample_db,inttostr(High(IconArr)));
  for i:=0 to High(IconArr) do
    begin
    writeln(sample_db,inttostr(IconArr[i].IconType));
    writeIcon(IconArr[i].Icon);
    writeln(sample_db,'');
    end;
  closefile(sample_db);

  //recognition_run:=true;

  }
end;

// відновлення картинки для збереження семпла
procedure TAddSampleForm.Button2Click(Sender: TObject);
var canvas:tcanvas;
begin
if debug then
begin
      canvas:=AddSampleForm.cellImage.canvas;
      Canvas.Brush.Color := ClWhite;Canvas.FillRect(Canvas.ClipRect);
      Canvas:=AddSampleForm.ObjectImage.Canvas;
      AddSampleForm.ObjectImage.Height :=0;
      AddSampleForm.ObjectImage.Width :=0;

end;
upreview.BitmapGrabEventArr[mainform.CamsListBox.ItemIndex].NeedAddSample :=
true;
end;

procedure TAddSampleForm.Cognitive_graphic_BP1EpochPassed(Sender: TObject);
begin
  prbEpoch.Position := prbEpoch.Position + 1;
  sttError.Caption := FloatToStr(Cognitive_graphic_BP1.TeachError);
  Application.ProcessMessages;

end;


// навчання нейромережі із зазначеними параметрами
procedure TAddSampleForm.Button3Click(Sender: TObject);
begin


      Cognitive_graphic_BP1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
      Cognitive_graphic_BP1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
      Cognitive_graphic_BP1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
      Cognitive_graphic_BP1.Init;
 // Учимо багатошарову
 Cognitive_graphic_BP1.EpochCount := speEpochCount.Value;
 prbEpoch.Max := Cognitive_graphic_BP1.EpochCount;
```

```
 prbEpoch.Position := 0;

 // Запуск процесу навчання (offline)
 //Cognitive_graphic_Extended1.TeachOffLine;
 Cognitive_graphic_BP1.TeachOffLine;
end;


// перебір параметрів навчання, навчання, записати помилки в лог
// для дослідження  - яка нейромережа краще, і як не потрапити в локальний
// мінімум при навчанні.
procedure TAddSampleForm.Button4Click(Sender: TObject);
  var log:textfile;
  var i,j:integer;
  var cnt:integer;
begin


  Cognitive_graphic_BP1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
  Cognitive_graphic_BP1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
  Cognitive_graphic_BP1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
  Cognitive_graphic_BP1.EpochCount := 5000;


  prbEpoch.Max := Cognitive_graphic_BP1.EpochCount;
  prbEpoch.Position := 0;
      assignfile(log,neuro_teach_log_file_name);
      append(log);
      writeln(log,'--------------------------');
      closefile(log);
  cnt:=400;
  for i:=1 to 20 do
    begin
    Cognitive_graphic_BP1.Alpha:=Cognitive_graphic_BP1.Alpha-0.01;
    for j:=1 to 20 do
      begin
      prbEpoch.Position := 0;
      Cognitive_graphic_BP1.TeachRate:=Cognitive_graphic_BP1.TeachRate-0.005;
      Cognitive_graphic_BP1.TeachOffLine;
      assignfile(log,neuro_teach_log_file_name);
      append(log);

      writeln(log,'помилка='+floattostr(Cognitive_graphic_BP1.TeachError)+'
альфа='+floattostr(Cognitive_graphic_BP1.Alpha)+'
шаг навчання =',floattostr(Cognitive_graphic_BP1.TeachRate));
      closefile(log);
      cnt:= cnt-1;
      sttError.Caption := inttostr(cnt);
      //prbEpoch.Position := prbEpoch.Position + 1;
      Application.ProcessMessages;
      end;
    end;

end;

// подія кінця епохи навчання , зрушення процес бара, відображення помилки
procedure TAddSampleForm.Cognitive_graphic_Extended1EpochPassed(Sender:
TObject);
begin
  prbEpoch.Position := prbEpoch.Position + 1;
  sttError.Caption := FloatToStr(Cognitive_graphic_Extended1.TeachError);
  Application.ProcessMessages;

end;


// дозвіл розпізнавання
procedure TAddSampleForm.Button5Click(Sender: TObject);
begin
```

```
  debug:=false;
  recognition_run:=not(recognition_run);


  if recognition_run then
    begin
    RobotRecognition := TRobotRecognition.Create(false);
    RobotRecognition.RecType:=AddSampleForm.recType.ItemIndex;
    AddSampleForm.Button5.Caption:='СТОП'
    end
    else
    begin
    //RobotRecognition.
    RobotRecognition.Terminate;
    //RobotRecognition.Destroy;
    AddSampleForm.Button5.Caption:='ПІШОВ!';
    end;

end;



// збереження образів у файл
procedure TAddSampleForm.Button6Click(Sender: TObject);
  var i,j:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff_file:textfile;
    max_icon_type:integer;
begin
  //IconTypes:=strtoint(eIconTypes.text);


  assignfile(sample_db,sample_db_file_name);
  rewrite(sample_db);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,inttostr(IconTypes));
  writeln(sample_db,inttostr(High(IconArr)));
  for i:=0 to High(IconArr) do
    begin
    writeln(sample_db,inttostr(IconArr[i].IconType));
    writeIcon(IconArr[i].Icon);
    writeln(sample_db,'');
    end;
  closefile(sample_db);

  //recognition_run:=true;


end;


// збереження ваг нейромережі у файл
procedure TAddSampleForm.Button7Click(Sender: TObject);
var

  neuro:textfile;
  i,j,w:integer;
begin
  AssignFile(neuro,neuro_weight_file_name);
  rewrite(neuro);
  writeln(neuro,floattostr(Cognitive_graphic_BP1.TeachRate));
  writeln(neuro,floattostr(Cognitive_graphic_BP1.Momentum));
  writeln(neuro,floattostr(Cognitive_graphic_BP1.Alpha));

  for i:=0 to Cognitive_graphic_BP1.LayerCount-1 do
    begin
    for j:=0 to Cognitive_graphic_BP1.Layers[i].NeuronCount-1 do
```

```
      begin
        for w:=0 to
length(Cognitive_graphic_BP1.Layers[i].Neurons[j].FWeights)-1 do
          begin

writeln(neuro,floattostr(Cognitive_graphic_BP1.Layers[i].Neurons[j].Weights[w]))
;
          end;
      end;
    end;
  closefile(neuro);
end;



// завантаження ваг нейромережі з файла
procedure TAddSampleForm.Button8Click(Sender: TObject);
var

  neuro:textfile;
  i,j,w:integer;
  buff:string;
begin
  AssignFile(neuro,neuro_weight_file_name);
  reset(neuro);


  readln(neuro,buff);Cognitive_graphic_BP1.TeachRate:=StrToFloat(buff);
  readln(neuro,buff);Cognitive_graphic_BP1.Momentum:=StrToFloat(buff);
  readln(neuro,buff);Cognitive_graphic_BP1.Alpha:=StrToFloat(buff);
  Cognitive_graphic_BP1.Init;
  for i:=0 to Cognitive_graphic_BP1.LayerCount-1 do
    begin
    for j:=0 to Cognitive_graphic_BP1.Layers[i].NeuronCount-1 do
      begin
        for w:=0 to
length(Cognitive_graphic_BP1.Layers[i].Neurons[j].FWeights)-1 do
          begin
            readln(neuro,buff);

Cognitive_graphic_BP1.Layers[i].Neurons[j].Weights[w]:=strtofloat(buff);
          end;
      end;
    end;
  closefile(neuro);

end;

procedure TAddSampleForm.Button9Click(Sender: TObject);
var
  QP: TQuickPixels;
  BitmapArr: TByteArr;
  y: Cardinal;

begin
  debug:=true;
 // upreview.BitmapGrabEventArr[mainform.CamsListBox.ItemIndex].NeedAddSample :=
true;
  QP := TQuickPixels.Create;
  QP.Attach(SampleImage.Picture.Bitmap);


  SetLength(BitmapArr,QP.Height);
  for y := 0 to High(BitmapArr) do
    SetLength(BitmapArr[y],QP.Width);

  ConvertBitmapToMonoChrome(QP,BitmapArr);

  //ConvertBitmapToMonoChrome(QP2,BitmapArr);
```

```
  if length(BitmapArr)>0 then
  segment(BitmapArr);
  debug:=false;
end;

procedure TAddSampleForm.TrackBar1Change(Sender: TObject);
begin
AddSampleForm.param.Caption:=inttostr( 100-AddSampleForm.TrackBar1.Position);
end;

procedure TAddSampleForm.Timer1Timer(Sender: TObject);
begin
AddSampleForm.lFPS.Caption:=floattostr(round(tm_tick/5*10)/10);
tm_tick:=0;
end;

end.
```

Файл Picture_Analyz_Hamming.pas - модуль розпізнавання графічних образів на знімках із супутників з використанням когнітивної графіки

```
unit Picture_Analyz_Hamming;
// один з модулів розпізнавання образів, містить:
// -повну функція сегментації й допоміжні процедури
// -підготовку й запуск розпізнавання нейромережею
// -процентне розпізнавання й супутні функції


interface

uses Graphics,SysUtils,Math,UTypeConst,Windows,UQPixels,NeuralBaseComp,Classes,
NeuralBaseTypes;

const
  Icon = 0;
  IconNot = 1;
  min_size=10;

type

  TByteArr = array of array of integer;

  TRobotRecognition = class(TThread) //Потік розпізнання
    private


    public
      //BitmapArr: TByteArr;
      BitmapReady:boolean;
      RecType:integer; // вибір параметрів розпізнання
      procedure Execute(); override; //запуск потоку
  end;


  TBitmap = Graphics.TBitmap;


  //масив байт Nx
  //TByteArr = array of array of byte;


  //семпл
  TIcon = record
    Icon: TByteArr;
    IconName:string;
    IconType: Cardinal;

  end;


type
    //Tlables =array of byte;
    Tlables =array of integer;
    Tlables_cnt=array of word;
    // інформація про сегменти образів
    Tsegments = record
    x:integer; // геометричне положення сегмента
    y:integer;
    top:integer;
    buttom:integer;
    left:integer;
    right:integer;
    segment_type:integer; // тип об'єкта - присвоюється підпрограмою розпізнання
    size:integer;// площа об'єкта (для фільтра)
    end;
```

```
    //масив іконок
    TIconArr = array of TIcon;
    TsegmentArr=array of Tsegments;
var
  BitmapArr: TByteArr;
  RobotRecognition:TRobotRecognition;
  leter_types:array [0..6] of char;
  sort_segments:Tsegments;
  segments:TsegmentArr;
  IconArr: TIconArr; //масив семплів
  //параметри перетворення в семпл
  IconSize: Cardinal = 16;
  IConSize: Cardinal = 16;
  MinBPixelsPercent: Cardinal = 80;
  IconTypes:Cardinal = 0;
  F: TextFile;
  left,right,top,buttom:integer;
  neuro_answer:integer;
  lowerest:integer;

procedure segment(BitmapArr: TByteArr);

procedure reader(letters_count:integer);

procedure ConvertBitmapToMonoChrome(var QPSource: TQuickPixels; var BitmapArr:
TByteArr);

//процедура перетворить масив BitmapArr Nx байт (монохромний бітмап) у масив
//IconAtrr[0..IconSize - 1][0..IConSize - 1], розбиваючи на осередки масив
//BitmapArr, MinBPixelsPercent - мінімальне відсоток чорних пікселів в осередку,
//достатніх, щоб зробити чорним елемент меншого масиву IConArr
procedure BitmapToIcon(var BitmapArr: TByteArr; var Icon: TByteArr;
                       IconSize:Cardinal; IconSize: Cardinal;
                       MinBPixelsPercent: Cardinal);

// функція порівнює два масиви байт Nx і видає відсоток співпавших байт
function CompareIcons(Arr1: TByteArr; Arr2: TByteArr): Cardinal;

//додати семпл зазначеного типу
procedure AddSample(var BitmapArr: TByteArr; IconType: Cardinal);

//семпл букви Т рулить
function TSampleRule(var BitmapArr: TByteArr): boolean;

implementation

uses
  Main_Cognitive_graphic_,Picture_processing_Hamming,upreview;



// головна функція нитки розпізнавання образів:
// розпізнає весь екран цілком, як один образ
// або б'є по сегментах, і розпізнає кожний окремо

// Вона сама просить зробити їй масив бітмепа на наступній події захоплення
// зображення камерою, як тільки обробить  попередні дані
// тут же вважається й FPS
procedure TRobotRecognition.execute();
begin
  FreeOnTerminate := true;
while not(RobotRecognition.Terminated) do
begin
  if BitmapReady then
    begin

      recognition_run:=false;
      //debug:=false;
      BitmapReady:=false;
```

```
      if rectype =1 then
      begin
        segment(BitmapArr);

      end;
      if rectype =0 then
      begin

        TSampleRule(BitmapArr);

      end;
              tm_tick:=tm_tick+1;
    recognition_run:=true;
    end;
end;
end;

procedure ConvertBitmapToMonoChrome(var QPSource: TQuickPixels; var BitmapArr:
TByteArr);
var
  i,x,y: Integer;
  Pixel: Cardinal;
  Pixel, Pixel, Pixel: byte;
  PixelRes,oldPix,oldPixRes: Cardinal;
  oldPixSum:real;
  oldPixCnt:integer;
  Canvas: TCanvas;

begin

oldPixCnt:=0;
OldPix:=0;
oldPixRes:=1;

  Canvas:=AddSampleForm.SampleImage.Canvas;


  left:=QPSource.Width - 1;
  right:=0;
  top:=QPSource.Height - 1;
  buttom:=0;

  for y := 0 to QPSource.Height - 1 do
  begin

    for x := 0 to QPSource.Width - 1 do
    begin
      Pixel := QPSource.GetPixels24(x,y);
      {
      Pixel := Pixel shr 23;
      Pixel := Pixel shr 15;
      Pixel := Pixel shr 7;
      }
      Pixel := Pixel shr 23;
      Pixel:= Pixel shr 15;
      Pixel := Pixel shr 7;
      //PixelRes := (Pixel and Pixel) and Pixel;
      PixelRes := Pixel;// and Pixel ;

      BitmapArr[y,x] := PixelRes;

     if PixelRes=1 then
      begin
        if x>right then right:=x;
        if y>buttom then  buttom:=y;
        if x<left  then left:=x;
        if y<top  then top:=y;
      end;
```

```
      // що- те на подобі простенького фільтра

      //BitmapArr[y,x] := PixelRes and oldPixRes;

      //BitmapArr[y,x] := round(oldPixSum) and 1;
      if debug then if BitmapArr[y,x]<>OldPix then canvas.Pixels[x,y]:=ClRed;

      //oldPixSum:=abs(oldPixSum+( PixelRes-OldPix)/10);

      OldPix:=PixelRes;


      //oldPixRes:=round((oldPixRes+PixelRes)/10);
      //oldPixRes:=PixelRes;

      //QPDest.SetPixels1(j,i,PixelRes {* clWhite});
    end;

  end;


end;


// обчислення даних нейромережею
// і зняття даних
procedure NeuroCompute(Arr1: TByteArr);
  var
  x,y: Integer;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
  cnt:integer;
  totle_layers,tmp,i:integer;
  true_exit:boolean;
  buff:string;
  Param,Param2:integer;

begin

  // вхідний параметр зняття даний з нейромережі
  Param:= 100-AddSampleForm.TrackBar1.Position;
  Param2:=AddSampleForm.TrackBar1.Position;

  Width := Length(Arr1[Low(Arr1)]);
  Height := Length(Arr1);


  // робимо вхідний вектор нейромережі з іконки (у модулів такий формат)
  cnt:=0;
  for x := 0 to Width - 2 do
    begin
      for y := 0 to Height - 2 do
        begin
      // багатошаровий
      if Arr1[y,x]=1 then  xInputVector[cnt] := 0 else xInputVector[cnt] := 1;
      //xInputVector[cnt] := Arr1[y,x];
      //Нейрона мережа Хопфілда
      //addsampleform.Cognitive_graphic_Hopf1.Layers[1].Neurons[cnt].Output :=
Arr1[y,x];
            cnt:=cnt+1;
        end;

    end;

    neuro_answer:=0;
    // нейромережа робить прогін у прямому напрямку
    addsampleform.Cognitive_graphic_BP1.Compute(xInputVector);

    totle_layers:=AddSampleForm.Cognitive_graphic_BP1.LayerCount-1;
```

```
    buff:='';

    // знімаємо даний з нейромережі
    for i := 0 to length(xOutputVector)-1 do
    begin

//xOutputVector[i]:=AddSampleForm.Cognitive_graphic_BP1.LayersBP[tmp].Neurons[i]
.Output;

tmp:=round(AddSampleForm.Cognitive_graphic_BP1.LayersBP[totle_layers].Neurons[i]
.Output*100);



    // суть алгоритму така - якщо якийсь із вихідних нейронів більше якогось
    // числа (наприклад 0.9, а всі інші вектора менше 0.1 кожний
    // те тоді відповіддю вважається той нейрон, що 0.9
    // ці параметри задаються з форми плазуючої
    // числа 0.9 і 0.1 актуальні по експериментах, але некритично
    // збільшити до 0.8 і 0.2, далі буде розпізнавати всі підряд.

    // подумав ще небагато: можна взагалі шукати максимальна відповідь,
    // як це зроблено у відсотках = тоді буде краще небагато
    // хоча міняти параметр нижче 0.8 однаково небезпечно - значить щось не так
    // на вхід іде.
    if tmp>Param then neuro_answer:=i+1; // собственно 0.9
    buff:=buff+' '+inttostr(tmp)
    end;
    true_exit:=true;
    for i := 0 to length(xOutputVector)-1 do
    begin
    if (neuro_ answer-1)<>i then
      begin

tmp:=round(AddSampleForm.Cognitive_graphic_BP1.LayersBP[totle_layers].Neurons[i]
.Output*100);
           if tmp>Param2 then true_exit:=false; // а це 0.1
      end;
    end;
    if not(true_exit) then neuro_answer:=0; // якщо не зустріли нічого іншого
              // більше чим 0.1 те даємо відповідь про успішний розпізнання,
              // інакше говоримо про те, що цей образ провалився.

    //mainform.NeuroCount.Text:=inttostr(neuro_answer);

end;



// це один з інстументів переприсвоєння міток,
// друга частина написана в правилах

function get_parent(var lables:Tlables;var lables_cnt:Tlables_cnt;
new_cnt:integer;test:integer):integer;
begin
  if (lables[new_cnt]<>0)  then result:=get_parent(lables,lables_cnt,
lables[new_cnt],test)
  else result:= new_cnt;
end;


function normalize(var lables:Tlables;var lables_cnt:Tlables_cnt;
index_cnt:integer;new_cnt:integer):integer;
var i,j:integer;
begin
      if lables[new_cnt]<>0 then
//      if lables[index_cnt]
      //lables[index_cnt]<>0
```

```
//      if lables[index_cnt]<>0 then
normalize(lables,lables_cnt,lables[index_cnt],index_cnt);
      //lables[index_cnt]:=new_cnt;
//      lables[new_cnt]:=index_cnt;
      //lables_cnt[index_cnt]
      //lables_cnt[new_cnt]:=lables_cnt[new_cnt]+lables_cnt[index_cnt];
      //lables_cnt[index_cnt]:=0;
      {
      for i:=1 to index_cnt do
        if lables[i]<>0 then
          begin
          for j:=1 to index_cnt do
            begin
            if lables[lables[i]]<>0 then
              begin
              lables[i]:=lables[lables[i]]; // і перепризначуємо влучні
              end;
            end;
          end;
        }
//result:= lables;
end;

// сегментація масиву
// у коментариях: L,m -lables
// цей же алгоритм застосовується в матлабе BWLABEL,
// реалізація цілком може бути іншою BWLABEL

// по суті - прохід по рядах і присвоєння міток за правилами,
// зазначеним нижче, якщо мітка вже привласнена комусь а треба НЕЮ привласнити
// поточну, то пошук її батька, і присвоєння ім'я батька.
// так само захист від присвоєння батька на самого себе.

// У результаті роботи алгоритму - після першого ж проходу масиву
// всі мітки посилаються або один на одного або на батька - він посилається на 0
// нумерація міток починається з 2 (т.до споконвічно масив містить тільки 1 і 0)

procedure segment( BitmapArr: TByteArr);
var i,ii,j :integer;
    width,height,x,y:integer;
    index_cnt:integer;
    lables:Tlables; // мітки, що розставляються спочатку
    lables_cnt:Tlables_cnt; // площа міток
    real_lables:array of integer; // кожна мітка  - унікальний масив
    //debug:boolean;
    seg_debug:textfile;
    seg_debug_file_name:string;
    flag:boolean;
    middel:integer;

begin


  setlength(real_lables,0);
  seg_debug_file_name:='data\seg.txt';
  //debug:=true;
  index_cnt:=1;
  setlength(lables,2);
  setlength(lables_cnt,2);
  height:=length(BitmapArr);
  width:=length(BitmapArr[0]);


  // прохід по масиві
  // первинна установка міток по нижченаписаним правилах
  for i:=1 to height-2 do
    begin
    for j:=1 to width-2 do
      begin
```

```
        if BitmapArr[i,j]>=1 then
            begin

            //   0       0
            // 0 1 ->  0 L

            if  (BitmapArr[i, j-1]=0) and
                (BitmapArr[ i-1,j]=0) then
                  begin
                  setlength(lables,length(lables)+1);
                  setlength(lables_cnt,length(lables_cnt)+1);
                  index_cnt:=index_cnt+1;
                  lables[index_cnt]:=0;
                  lables_cnt[index_cnt]:=1;
                  BitmapArr[i,j]:=index_cnt;
                  end;

            //   0       0
            // L 1 ->  L L

            if  (BitmapArr[i, j-1]>1) and
                (BitmapArr[ i-1,j]=0) then
                  begin
                  BitmapArr[i,j]:=BitmapArr[i, j-1];
                  lables_cnt[BitmapArr[i, j-1]]:=lables_cnt[BitmapArr[i, j-
1]]+1;
                  end;

            //   L       L
            // 0 1 ->  0 L

            if  (BitmapArr[i, j-1]=0) and
                (BitmapArr[ i-1,j]>1) then
                  begin
                  //BitmapArr[i,j]:=get_parent(lables,lables_cnt,BitmapArr[ i-
1,j]);
                  BitmapArr[i,j]:=BitmapArr[ i-1,j];
                  lables_cnt[index_cnt]:=lables_cnt[index_cnt]+1;
                  //lables_cnt[BitmapArr[i,j]]:=lables_cnt[BitmapArr[i,j]]+1;
                  end;

            //   L       L
            //L 1 ->  L L

            if  (BitmapArr[i, j-1]>1) and
                (BitmapArr[i, j-1]=BitmapArr[ i-1,j]) and
                (BitmapArr[ i-1,j]>1) then
                  begin
                  BitmapArr[i,j]:=BitmapArr[ i-1,j];
                  lables_cnt[BitmapArr[i, j-1]]:=lables_cnt[BitmapArr[i, j-
1]]+1;
                  end;


            //   L       L
            // M 1 ->  M L (M:=L)
            // якщо L не вказує на 0, то пошук її самого далекого родича
            // якщо M це далекий родич L те не призначити M лінк на саму себе

            if  ((BitmapArr[i, j-1]>1) and
                (BitmapArr[i, j-1]<>BitmapArr[ i-1,j]) and
               //(lables[BitmapArr[i, j-1]]=0) and
                (BitmapArr[ i-1,j]>1)) then
                  begin
                  if (lables[BitmapArr[ i-1,j]]<>BitmapArr[i, j-1])then
                  begin
                  middel:=get_parent(lables,lables_cnt,BitmapArr[ i-
1,j],BitmapArr[i, j-1]);
```

```
                  if (BitmapArr[i, j-1]<>middel) then
                  begin
                  lables[BitmapArr[i, j-1]]:=middel;
                  end;
                  BitmapArr[i,j]:=middel;//BitmapArr[ i-1,j];
                  end
                  //set_parent(lables,lables_cnt,index_cnt,BitmapArr[ i-1,j]);
                  else
                  begin
                  BitmapArr[i,j]:=get_parent(lables,lables_cnt,BitmapArr[ i-
1,j],BitmapArr[i, j-1]);
                  end;

                  end;

            //   L       L   M<>0
            // M 1 ->  M L (L:=M)
            end; // if BitmapArr[i,j]=1 then

//            assignfile(seg_debug,seg_debug_file_name);
//            append(seg_debug);

//             write(seg_debug,inttostr(BitmapArr[i,j]));
//            closefile(seg_debug);
      end; // for j
//      assignfile(seg_debug,seg_debug_file_name);
//      append(seg_debug);

//      writeln(seg_debug,'');
//    closefile(seg_debug);
    end; //for i

    // отже нормалізуємо масив посилань лейблів один на одного
    // у результаті повинні  вийти набагато менше лейблів, але кожний буде
    // унікальним об'єктом, і піде в підпрограму розпізнання зі своїм номером
    if debug then
      begin
      assignfile(seg_debug,seg_debug_file_name);
      rewrite(seg_debug);
          for i:=1 to height-2 do
            begin
            for j:=1 to width-2 do
              begin
              //if BitmapArr[i,j]>1 then  write(seg_debug,inttostr(1)) else
write(seg_debug,inttostr(0))
              write(seg_debug,inttostr(BitmapArr[i,j]));
              end;
              writeln(seg_debug,'');
            end;
      writeln(seg_debug,'----------');

      for i:=1 to index_cnt do
        writeln(seg_debug,inttostr(i)+' '+inttostr(lables[i])+'
'+inttostr(lables_cnt[i]));
      writeln(seg_debug,'----------');

      end;


      for i:=1 to index_cnt do
        if lables[i]<>0 then
          begin
          for j:=1 to index_cnt do
            begin
            if lables[lables[i]]<>0 then
              begin

              lables[i]:=lables[lables[i]]; // і перепризначуємо влучні
              end;
```

```
            end;
          end;

    // можна сполучити з попереднім
    //
    // !!!!!!!!!!
    // на цьому ж етапі краще шукати границі кожного образа, що б потім
    // не проходити масив заданого стільки разів ,скільки образів знайдене

    // Плюси : кількість повних проходів скоротитися ґрунтовно:
    // Мінуси...м.. напевно їх немає.. так що треба буде зробити обов'язково.
    for i:=0 to  index_cnt do
      begin
        if lables[i]<>0 then
         begin
          lables_cnt[lables[i]]:=lables_cnt[lables[i]]+lables_cnt[i];
//підсумуємо ваги
         end;
      end;

       // Заглушка алгоритму нормалізації - тимчасово!
{
  for i:=1 to height-2 do
    begin
    for j:=1 to width-2 do
      begin
        if BitmapArr[i,j]>1 then
            begin
            if  (BitmapArr[i, j-1]>1) and
                (BitmapArr[i, j-1]<>BitmapArr[ i-1,j]) and
                //(lables[BitmapArr[i, j-1]]=0) and
                (BitmapArr[ i-1,j]>1) then
                  begin
                  BitmapArr[i,j]:=BitmapArr[ i-1,j];

                  //lables[BitmapArr[ i-1,j]]:=BitmapArr[i, j-1];
                  //lables_cnt[BitmapArr[i, j-1]]:=lables_cnt[BitmapArr[i, j-
1]]+1;
                  // спробуємо навпаки
                  lables[BitmapArr[i, j-1]]:=BitmapArr[ i-1,j];
                  lables_cnt[BitmapArr[ i-1,j]]:=lables_cnt[BitmapArr[ i-
1,j]]+1;

                  end;

            end;
      end;

 }


    if debug then
      begin
      for i:=1 to index_cnt do
        writeln(seg_debug,inttostr(i)+' '+inttostr(lables[i])+'
'+inttostr(lables_cnt[i]));
      writeln(seg_debug,'----------');

      end;

    // тепер зробимо прохід по масиві ще раз і перепишемо мітки
    // на нормалізовані, не хочеться витрачати час на ще один прохід

    for i:=1 to height-2 do
    begin
    for j:=1 to width-2 do
      begin
        if BitmapArr[i,j]<>0 then
          begin
```

```
            if lables[BitmapArr[i,j]]<>0 then
             begin

               BitmapArr[i,j]:=lables[BitmapArr[i,j]];

             end;
           end;

      end;
    end;

    if debug then
//        if true then
      begin
//             assignfile(seg_debug,seg_debug_file_name);
//      rewrite(seg_debug);

          for i:=1 to height-2 do
            begin
            for j:=1 to width-2 do
              begin
              write(seg_debug,inttostr(BitmapArr[i,j]));
              end;
              writeln(seg_debug,'');
            end;
//                              closefile(seg_debug);
      end;

    // SetLength(IconArr,Length(IconArr) + 1);

    // дивимося які мітки були унікальними
    if debug then
      begin
      write(seg_debug,'----дивимося які мітки були унікальними-');
      end;

      for i:=2 to index_cnt do
        if lables[i]=0 then
         begin
          if length(real_lables)>0 then
            begin
            // перевірка мітки на унікальність
            flag:=true;
            for j:=0 to (length(real_lables)-1) do
              begin
                if real_lables[j]=i then flag:=false;
              end;
              // якщо так, те унікальна
              if flag then
                begin
                  setlength(real_lables,length(real_lables)+1);
                  real_lables[length(real_lables)-1]:=i;
                end;
            end
            else
            begin
              setlength(real_lables,length(real_lables)+1);
              real_lables[length(real_lables)-1]:=i;
            end
         end;

      if debug then
         begin

         writeln(seg_debug,'--Унікальні мітки--');
           for i:=0 to length(real_lables)-1 do
           begin
             writeln(seg_debug,inttostr(real_lables[i])+'
'+inttostr(lables_cnt[real_lables[i]]));
```

```
       end;


     end;


// ну от, переходимо до суті - властиво до розпізнавання образів:
// на цьому етапі вже є матриця, у якій всі помічено не одиничками,
// а цифрами, до якому сигменту все це ставиться

// тепер треба чи подивитися достатня площа об'єкта
// (або навпаки занадто великувата)
// і послати його розпізнаватися, заодно заготовивши структуру з його даними


char_cnt:=1;
setlength(segments,0);
j:=0;
i:=0;
if length(real_lables)>0 then
begin
for i:=0 to length(real_lables)-1 do
 begin
  if  lables_cnt[real_lables[i]]>min_size then
  begin

        left:=Width - 2;
        right:=0;
        top:=Height - 2;
        buttom:=0;

      for y := 0 to height-2 do
        begin

        for x := 0 to Width - 2 do
          begin

          if BitmapArr[y,x]=real_lables[i] then
            begin
            if x>right then right:=x;
            if y>buttom then  buttom:=y;
            if x<left  then left:=x;
            if y<top  then top:=y;
            end;
          end;
        end;

  //writeln(seg_debug,inttostr(real_lables[i]));
  //top:=0;buttom:= height-2;left:=0;right:= width-2;

  if (right<>0) and (buttom<>0) then
  begin
    setlength(segments,(length(segments)+1));


    // пропускаємо через підпрограму розпізнання
    imFeatures(BitmapArr,real_lables[i]);

    // і так само пропускаємо  через нейромережу
    NeuroCompute(BitmapArr3);
    segments[j].x:=segment_X;
    segments[j].y:=segment_Y;
    segments[j].size:=lables_cnt[real_lables[i]];
    segments[j].segment_type:=neuro_answer;
    j:=j+1;
  end;
  end;
 end;
```

```
     // що -те робимо з типами образів, наприклад читаємо або
     // виставляємо прапорці для інший програми робота.
     if (( j-1)>=0)and(j=length(segments)) then   reader( j-1);

     end;



    // дебаг закінчився
    if debug then
      begin
      closefile(seg_debug);
      end;

end;


procedure reader(letters_count:integer);
var i,j,ii:integer;
begin
          // прочитаний текст - для початку опустошаємо усе з попереднього кроку
          AddSampleForm.reader.Text:='';
         // сортуємо букви
{    if letters_count>1 then
    begin
    for i := letters_count downto 0 do
      begin
   //   if debug then writeln(F,inttostr(segments[i].x)+'
'+inttostr(segments[i].segment_type));
      for ii := 0 to i do
      if segments[ii].x > segments[ii+1].x then
      begin
       sort_segments := segments[ii];
       segments[ii] := segments[ii+1];
       segments[ii+1] := sort_segments;
       end;
      end;
    end;

 }

     for i:=0 to letters_count do
     begin
     if (segments[i].segment_type<length(leter_types))and
(segments[i].segment_type>0) then

AddSampleForm.reader.text:=AddSampleForm.reader.Text+leter_types[segments[i].seg
ment_type];
     end;


     if AddSampleForm.ComboBox2.ItemIndex=1 then
       SayText(AddSampleForm.reader.Text);

end;


//семпл букви Т рулить
// стара функція для розпізнання
// порівнює дві матриці (одна з екрана, інша з масиву еталонів)
// результат дає якщо кількість що збіглися пікселей більше якогось відсотка
// а так само якщо воно найбільше із всіх що збіглися

// по експериментах - нейромережа працює луше від 10% до 30% по цьому
// далі використовувати процентрое порівняння
// практично ні до чого
function TSampleRule(var BitmapArr: TByteArr): boolean;
var
  Icon: TByteArr;
```

```
  y,i,j: Integer;
  MaxCompareTPercent: Cardinal;
  MaxCompareNotTPercent: Cardinal;
  CurComparePercent: Cardinal;
  iconFind:integer;
begin
  Result := false;
  SetLength(Icon,IConSize);
  for y := 0 to IconSize - 1 do
    SetLength(Icon[y],IconSize);


    // якась заглушка.. чи не знаю потрібна чи зараз ні
    if  right<>0 then
  begin
  // перетворимо ВЕСЬ екран в іконку (з першого білого, до останнього білого
  // пікселя.
 imFeatures(BitmapArr,1);

  if length(BitmapArr3)>1 then
 Icon:=BitmapArr3;


  // для процентного порівняння
  MaxCompareTPercent := 0;
  MaxCompareNotTPercent := 0;
  CurComparePercent := 0;


  {
  writeln(F,'--Бітмап--');
  for i := 0 to IconSize - 1 do
  begin
    for j := 0 to IconSize - 1 do
      write(F,Icon[i,j]);
    writeln(F);
  end;
  writeln(F);
  }

  //Порівнюємо сіткою
  NeuroCompute(Icon);


  // порівнюємо по пікселям що співпали
  iconFind:=0;
  for y := 0 to High(IconArr) do
  begin
    CurComparePercent := CompareIcons(Icon,IconArr[y].Icon);
    {
    if (IconArr[y].IconType = Icon) and (CurComparePercent > MaxCompareTPercent)
then
      MaxCompareTPercent := CurComparePercent;
    if (IconArr[y].IconType = IconNot) and (CurComparePercent >
MaxCompareNotTPercent) then
      MaxCompareNotTPercent := CurComparePercent;
    }
     if CurComparePercent > MaxCompareTPercent then
     begin
     MaxCompareTPercent := CurComparePercent; iconFind:=IconArr[y].IconType;
     end;

  end;
  if MaxCompareTPercent < 80 then iconFind:=0;
  //MainForm.Caption := IntToStr(iconFind);
  //MainForm.Caption := IntToStr(MaxCompareTPercent);
  //MainForm.Caption := '-no-';
//  if (MaxCompareTPercent > MaxCompareNotTPercent) and (MaxCompareTPercent >
80) then
```

```
 //   begin {MainForm.Caption := 'OK';} Result := true; end;
 end;
end;

// генеруємо іконку заданого розміру з масиву
procedure AddSample(var BitmapArr: TByteArr; IconType: Cardinal);
var
  Icon: TByteArr;
  x,y: Integer;
  canvas:tcanvas;
begin
  SetLength(IconArr,Length(IconArr) + 1);

  SetLength(Icon,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(Icon[y],IconSize);

  SetLength(IconArr[High(IconArr)].Icon,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(IconArr[High(IconArr)].Icon[y],IconSize);

  //BitmapToIcon(BitmapArr,Icon,IconSize,IconSize,MinBPixelsPercent);

  //1canvas:=AddSampleForm.cellImage.canvas;

  for y := 0 to High(Icon) do
    for x := 0 to High(Icon[Low(Icon)]) do
    begin

      IconArr[High(IconArr)].Icon[y,x] := BitmapArr3[y,x];
      if Icon[y,x]=1 then


    end;

  IconArr[High(IconArr)].IconType := IconType;
end;

// функція порівнює два масиви байт Nx і видає відсоток співпавших байт
function CompareIcons(Arr1: TByteArr; Arr2: TByteArr): Cardinal;
var
  x,y: Integer;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
  cnt:integer;
begin
  SamePixels := 0;
  Width := Length(Arr1[Low(Arr1)]);
  Height := Length(Arr1);


//  cnt:=0;

  if  debug  then
  begin
  for x := 0 to Width - 2 do
    begin
      for y := 0 to Height - 2 do
         write(F,Arr1[y,x]);
      write(F,' ');
      for y := 0 to Height - 2 do
        write(F,Arr2[y,x]);
      writeln(F,' ');
    end;
        writeln(F,' ');
  end;
  for y := 0 to Height - 2 do
    for x := 0 to Width - 2 do
    begin
```

```
      if Arr1[y,x]=1 then
      //xInputVector[cnt] := 1
      //Нейрона мережа Хопфілда
      //addsampleform.Cognitive_graphic_Hopf1.Layers[1].Neurons[cnt].Output := 1
      else
      //xInputVector[cnt] := 0;
      // Нейрона мережа Хопфілда
      //addsampleform.Cognitive_graphic_Hopf1.Layers[1].Neurons[cnt].Output := -
1;

      cnt:=cnt+1;

      if Arr1[y,x] = Arr2[y,x] then
        SamePixels := SamePixels + 1;
    end;
    //addsampleform.Cognitive_graphic_Hopf1.Calc;
    Result := Round(SamePixels * 100 / (Width * Height));
end;


// Ця процедура використовувалася раніше
// зараз уже не потрібна! - ніде не викликається

//процедура перетворить масив BitmapArr Nx байт (монохромний бітмап) у масив
//IconAtrr[0..IconSize - 1][0..IConSize - 1], розбиваючи на осередки масив
//BitmapArr, MinBPixelsPercent - мінімальне відсоток чорних пікселів в осередку,
//достатніх, щоб зробити чорним елемент меншого масиву IConArr
procedure BitmapToIcon(var BitmapArr: TByteArr; var Icon: TByteArr;
                           IconSize: Cardinal ; IconSize: Cardinal ;
                           MinBPixelsPercent: Cardinal );


var
  i,j,x,y,bx,by,ix,iy: Integer; //Лічильники
  CSize, CSize: Integer; //Розміри осередку
  BPorog: Cardinal; //поріг чорного
  BPixelsInCell: Cardinal; // кількість чорних пікселей в осередку
  xLeft, xRight, yTop, yBottom : integer;          // абс. коорд. образа
  line:string;
  const color:integer = 0;
begin


  // перетворимо трохи пікселей на ділянці в один осередок, зчитуємо колір
  CSize := Floor(Length(BitmapArr) / IconSize);
  CSize := Floor(Length(BitmapArr[Low(BitmapArr)]) / IconSize);

  Bporog := Round(CSize * CSize / 100 * MinBPixelsPercent);

  bx := 0;
  by := 0;
  BPixelsInCell := 0;

  for iy := 0 to IconSize - 1 do
    for ix := 0 to IconSize - 1 do
    begin
      for y := by to by + CSize - 1 do
        for x := bx to bx + CSize - 1 do
          if BitmapArr[y,x] = 0 then
            BPixelsInCell := BPixelsInCell + 1;
      if BPixelsInCell > BPorog then
        Icon[iy,ix] := 0
      else
        ICon[iy,ix] := 1;
//        line:=line+inttostr(ICon[iy,ix]);
      BPixelsInCell := 0;
      bx := bx + CSize;
      if Round(bx / CSize) >= IConSize then
      begin
```

```
        bx := 0;
        by := by + CSize;
      end;

    end;

end;

initialization
  AssignFile(F,'Sampledebug.txt');
  Rewrite(F);
  leter_types[0]:=' ';
  leter_types[1]:='П';
  leter_types[2]:='И';
  leter_types[3]:='М';
  leter_types[4]:='Л';
  leter_types[5]:='О';
  leter_types[6]:='Д';

finalization
  CloseFile(F);


end.
```

Файл About.pas - довідка

```
unit about;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TAboutForm = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutForm: TAboutForm;

implementation

{$R *.dfm}

procedure TAboutForm.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('МАГІСТЕРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Дослідження та програмна реалізація системи розпізнавання
графічних образів на знімках із супутників з використанням когнітивної
графіки');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Лисенко І.А.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Пойченко Олександр Петрович');
Memo1.Lines.Add('                    гр. КН-22МЗ');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2023');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
end;

procedure TAboutForm.Button1Click(Sender: TObject);
begin
AboutForm.Close;
end;

end.
```