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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21-3СК
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Павлюхін В.Л.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*“*Інформаційні технології*”*_____
Спеціальність *125 “*Кібербезпека*”*
Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Павлюхіну Владиславу Леонідовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 136-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи кібербезпеки в промислову експлуатацію.
6. Висновки

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

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Павлюхін В.Л.
(прізвище та ініціали)

АНОТАЦІЯ

Павлюхін В.Л. Програмне забезпечення системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону.

Метою розробки є програмне забезпечення системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону.

Результат роботи – програмна реалізація системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону.

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 мобільному пристрої під керуванням ОС Android.

Програму розроблено в середовищі Delphi 10.4.1.

Ключові слова: кібербезпека, біометрія

ABSTRACT

Pavliukhin V.L. Software of the cyber security system of biometric identification of the face of the smartphone user. 125 Cyber security. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the cyber security system of biometric identification of the face of the smartphone user.

The purpose of the development is the software of the cyber security system of biometric identification of the face of the smartphone user.

The result of the work is the software implementation of the cybersecurity system of biometric identification of the face of the smartphone user.

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 a mobile device running the Android OS.

The program was developed in the Delphi 10.4.1 environment.

Keywords: cyber security, biometrics

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ВО	–	виявлення облич
ЗНМ	–	згорткова нейронна мережа
ККНК	–	комбінований каскад нейромережних класифікаторів
КСК	–	каскад слабких класифікаторів
РО	–	розпізнавання облич

ВСТУП

Актуальність теми. За останні п'ятдесят років нестримне прискорення технологічного розвитку повністю змінило суспільство. Гіперзв’язок, створений завдяки розгортанню комунікаційних мереж, дозволяє оцифрувати значну частину нашої діяльності, приносячи численні переваги, а також деякі ризики, які необхідно вирішити. У цій аналогово-цифровій подвійності одним із процесів, який залишається ключовим для забезпечення безпеки, є перевірка особи.

У фізичному світі це завдання, яке ми виконуємо щодня інтуїтивно, впізнаючи людей з першого погляду. Коли ця сама (аж ніяк не тривіальна) діяльність повинна здійснюватися автоматично без втручання людини, необхідно задіяти весь технологічний потенціал для отримання надійних результатів.

У цьому сенсі біометрія в поєднанні з технологіями штучного інтелекту формує надійну команду, яка дозволяє нам забезпечувати безпеку цифрових процесів, де необхідно гарантовано підтвердити особу людини.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону.

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

– Огляд існуючих систем біометричної ідентифікації обличчя користувача смартфону.

– Дослідження системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону.

– Програмна реалізація системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі біометричної ідентифікації обличчя користувача смартфону.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для біометричної ідентифікації обличчя користувача смартфону. Інтерес до всіляких систем біометричної ідентифікації стрімко росте. Опитування жителів в 40 країн миру показав, що 52% хочуть захистити свій смартфон і замість пароля використовувати відбиток пальця, і ще 48% мріють про системи ідентифікації райдужної оболонки ока для розблокування пристрою (дані Ericsson). Протягом п'яти років щорічне число відвантажень смартфонів і планшетів, що реалізують розглянуті біометричні функції, збільшиться до 665 млн. (дані ABI research).

До таких споживчих настроїв активно підлаштовуються виробники мобільних пристроїв і витрачають на технології безпеки усе більше засобів. Дослідники з Visiongain підрахували, що світовий ринок мобільної безпеки до кінця 2024 року досягне $3,5 млрд. В Biometrics Research Group (BRG) прогнозують, що наступні покоління смартфонів будуть включати біометрію. "Інтеграція буде обумовлена великими виробниками", – припустили аналітики BRG.

1.2 Область застосування

Областю застосування є системи біометричної ідентифікації. Виробники, прагнучи виділитися із загальної маси смартфонів, активно розробляють і впроваджують біометричні системи, засновані на скануванні частин тіла, голосу й навіть ходи. Приведемо п'ять найвідоміших технологій розпізнавання.

За відбитком пальців

Компанія Motorola одна з перших почала використовувати біометричні дані в пристроях. В 2021 році компанія випустила смартфон Motorola Atrix зі сканером відбитків пальців. Але самим нашумілим пристроєм, що впровадив біометрію, став iPhone 5S зі сканером Touch ID. Розробкою Apple зробив фурор на презентації смартфона 10 вересня 2023 року. За перші вихідні компанія продала рекордні 9 млн. смартфонів.

Нову систему ідентифікації розкритикували багато фахівців з безпеки. Найшлися ентузіасти, які зібрали винагороду для хакера, що зможе її зламати. Призовий фонд для зломщика склав $16 тис. Перемога дісталася групі німецьких хакерів. Протягом місяця вони придумали як обійти захист. Для цього їм знадобилося сфотографувати відбиток пальця власника iPhone зі скляної поверхні.

По шляху Apple пішла HTC, показавши власний смартфон, оснащений сканером відбитків пальців. Але якщо в iPhone 5S сканер убудований у кнопку, то в HTC Max One він розташований на задній панелі.

По сітківці ока

Ходять упорні слухи, що новий Galaxy S5 від Samsung буде відрізнятися не тільки могутнішим процесором, камерою, водонепроникним корпусом і розміром екрана.

Корейці вирішили піти далі своїх американських конкурентів з Apple, і оснастити свій флагман сканером сітківки ока.

Райдужна оболонка, як і відбитки пальців, має свій унікальний малюнок. При цьому сканування ока відбувається швидше й точніше, навіть якщо користувач носить окуляри або контактні лінзи.

За особою

На початку грудня 2023 року Apple одержала патент на власну технологію розпізнавання осіб. Технологія порівнює риси й форму особи, колір шкіри й інші показники із зафіксованими параметрами власника.

Передбачається, що застосовувати технологію не тільки для ідентифікації при включенні смартфону, але й при використанні деяких функцій. Наприклад, одержати інформацію про того, хто дзвонить, можна тільки тоді, коли смартфон розпізнав власника. Схожі системи використовуються в цей час Android, PlayStation 4 і Xbox One.

За голосом

Торік компанія Nuance Communications, відома по розробці сервісу по розпізнанню голосу Siri, показала технологію, що дозволяє знімати блокування смартфону за голосом користувача. Технологія використовує відразу два рівні захисту: біометричний, котрий розпізнає ваш унікальний "відбиток голосу", і механізм із паролем або кодовою фразою, які потрібно вимовити.

Аналітики не виключають, що технологією може зацікавитися Apple. Глава компанії Тім Кук заявляв про необхідність обновити функціонала Siri.

За ходою

Розроблювачі перебувають у пошуку нових біометричних технологій ідентифікації. Група вчених з Норвегії й Німеччини придумали механізм перевірки користувача за допомогою акселерометра, убудованого в смартфон, пише MIT Technology Review.

Розпізнавати користувача пропонується за ходою, оскільки саме її вчені вважають індивідуальною характеристикою людини.

Незважаючи на те, що рівень погрішності в розпізнаванні досить високий (20%), систему можна буде використовувати в сполученні з іншими способами автентифікації.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Значний інтерес до біометричних рішень проявляють світових гігантів – наприклад, Citigroup і McDonald's. Країни Шенгенської зони оголосили про плани введення біометричних віз. Нові технології й засоби біометричної ідентифікації як і раніше активно впроваджуються в аеропортах різних країн миру. Нові замовники із числа фінансистів, ритейлірів, керівників навчальних закладів, юридичних фірм, установ охорони здоров'я теж оцінили для себе переваги біометрії.

Як показує дійсність, традиційні методи персональної ідентифікації, засновані на застосуванні паролів або матеріальних носіїв, таких як пропуск, паспорт, водійське посвідчення, електронний ключ або карта, уже не відповідають сучасним вимогам безпеки. Пароль можна забути або перехопити, матеріальний носій – скопіювати, втратити або передати іншій особі. Саме це не дозволяє традиційним системам контролю доступу забезпечувати належний рівень надійності, що приводить до істотних фінансових втрат. От чому усе більше компаній шукають більше ефективні методи забезпечення безпеки, звідси й тенденція переходу до біометричних систем ідентифікації, тобто таким системам, де перевірка людини відбувається на основі унікальних біометричних особливостей кожного конкретного індивідуума.

Ідентифікація особистості за допомогою біометричних технологій є одним із самих перспективних і бурхливо, що розвиваються напрямків, причому провідну позицію серед біометричних методів і засобів визначення особистості об'єктивно займають дактилоскопічні системи – ідентифікація за відбитками

пальців. Системи, засновані на розпізнаванні райдужної оболонки ока або візерунка кровоносних посудин на сітківці ока, набагато більше складні у використанні й коштовні. Це ж стосується й досить малонадійних систем, заснованих на розпізнаванні фото- або відеообраза особи.

Біометрична інформація є унікальною (навіть в ідентичних близнюків відбитки пальців розрізняються), вона не може бути забута або загублена, і, крім того, для пред'явлення такої інформації потрібна фізична присутність її носія, тобто самої людини. Тому біометричний компонент стає фактично обов'язковим елементом сучасних систем контролю доступу, що висувають підвищені вимоги до надійності.

Технологія верифікації за відбитками пальців

Найбільш популярною біометричною технологією є дактилоскопія – ідентифікація за відбитками пальців.

Консалтингова компанія Acuity Market Intelligence у своєму дослідженні наводить дані про те, що цю технологію використовують понад 70% компанії на світовому біометричному ринку, і в доступній для огляду перспективі (5-7 років) вони збережуть домінуюче положення. Друге місце зараз у технологій ідентифікації за особою (13,7%), однак не виключено, що надалі його завоюють системи ідентифікації за райдужною оболонкою очей, які з нинішніх скромних семи відсотків до 2027 р. можуть завоювати частку в 18,8%.

Популярність даної технології обумовлена оптимальним співвідношенням точності, надійності й швидкості ідентифікації із ціною, у яку обходиться закупівля біометричної системи її замовникові. У системах розглянутого типу ймовірність помилкової ідентифікації становить 0,000000001%, а час, необхідний для сканування відбитка, становить частки секунди. Вартість якісних оптичних сканерів відбитків пальців виміряється декількома десятками доларів, тоді як для сканерів райдужної оболонки ціни обчислюються в сотнях, а для нав'язливо, що пропагувалися систем, ідентифікації по тривимірній моделі черепа – тисячами й десятками тисяч доларів.

Біометрія на стражі банківського сектора й рітейлу

Сфери застосування біометричних систем можна розділити на дві основні групи: до першої відносяться підприємства, де конфіденційність і схоронність інформації мають першорядне значення й величезну цінність – це банки, страхові компанії, виробничі компанії. Головна проблема в забезпеченні інформаційної безпеки даних компаній – боротьба з інсайдерами. Одержавши облікові дані своїх колег, вони здатні оперувати величезними сумами й поставити під погрозу існування весь бізнес.

Друга група містить у собі підприємства, що потребують у якісному обліку робочого часу й контролю фізичного доступу в приміщення. Як правило, це великі роздрібні мережі, ресторани, готелі, заводи.

Пристрої біометричних систем

Переваги біометричних систем контролю доступу:

– Ідентифікація користувача відбувається за відбитками пальців, які є унікальними навіть для ідентичних близнюків.

– На відміну від паролів, смарт-карт і брелоків, відбитки пальців не можна украсти, підглянути або втратити. Безпечний пароль повинен містити мінімум 12 різних символів. Але його важко запам'ятати, а простій легко підібрати шахраям.

– Для захисту особливо важливої інформації й приміщень, біометричні системи можуть використовувати багатофакторну ідентифікацію по системі «відбиток пальця+пароль+ смарт-карта».

– Украсти відбиток з бази даних біометричної системи неможливо, у ній зберігається лише цифрова модель, яку не можна відновити.

– Імовірність помилкової ідентифікації користувача становить 0,000000001%.

– Біометричні системи швидко окупаються й не вимагають наступних витрат на дозакупівлю матеріальних носіїв (смарт-карт, брелоків).

Деякі ситуації, однак, начебто спеціально створюються для застосування біометричних пристроїв. Наприклад, якщо ваша служба підтримки завалена

запитами користувачів, що забули свої паролі (по оцінках компанії Gartner, вони становлять 40% всіх звернень до цієї служби), вам саме й варто звернутися до технології біометричної автентифікації.

Переваги й недоліки

Розробники всіх біометричних пристроїв висувають специфічні вимоги до програмних і апаратних засобів. Перевірте, чи є необхідні ресурси для підтримки вибраного пристрою й чи зможе цей пристрій працювати з мережним ПЗ. Крім того, з'ясуєте, потрібно й чи є в наявності зовнішнє джерело живлення або порт USB.

Усілякі страхи й культурні й релігійні забобони теж можуть працювати проти. Опитаєте службовців на предмет того, як вони сприймають ідею використовувати для автентифікації біометричні пристрої, і проведіть випробування пристрою, щоб довідатися, чи здатні вони (службовці) акуратно використовувати його.

І звичайно, "фахівці" уже знайшли способи обманювати біометричні пристрої. Відбитки пальців можна зняти з будь-якої гладкої поверхні, навіть прямо зі сканера відбитків пальців, за допомогою графітового порошку й шматка клейкої стрічки або желатину. Сканери райдужної оболонки нескладно обдурити, використовуючи фотографію ока користувача, зроблену з високим розрішенням. Щоб виявити обман, новітні пристрої реєструють "ознаки життя", зокрема пульсацію кровоносних посудин.

Установка порогів

Для біометричних пристроїв прийнятний поріг невдач у розпізнаванні встановлюється на основі відсотка помилкових дозволів на допуск (False Acceptance Rate – FAR) і відсотка помилкових відмов у допуску (False Rejection Rate – FRR). FAR відповідає ймовірності того, що біометричний пристрій помилковий визнає користувача, а FRR – що воно помилково відкине його.

Якщо адміністратор занижує поріг відмови в допуску, то система буде більше "поблажливо" оцінювати збіг збереженого в пристрої біометричного

зразка з даними користувача, і, природно, збільшиться ймовірність, що вона помилково дозволить вхід сторонньому. Установлюючи поріг занадто високо, ви збільшуєте ймовірність того, що система буде відкидати цілком легітимних користувачів. Щоб спростити експлуатацію системи, переконаєтеся, що пороги встановлюються й коректуються на місці.

Реєстрація й інтеграція

У будь-якій системі автентифікації користувачі спочатку повинні бути зареєстровані, тобто внесені в список допуску. Багато біометричних систем дозволяють самостійно робити це користувачам. Останні проходять автентифікацію на локальній машині або сервері довідника й потім реєструються за допомогою біометричного пристрою. На жаль, якщо ви застосовуєте біометричні пристрої для підвищення надійності автентифікації, але при первісній ідентифікації й автентифікації цілком покладаєтеся на імена й паролі користувачів, то ви не одержуєте ніяких переваг у плані захисту. Реєстрація користувачів, виконувана під контролем адміністратора, цю проблему вирішує, але вона займає більше часу.

Розібравшись із реєстрацією, визначитеся з тим, де будуть зберігатися біометричні дані автентифікації. Системи, що зберігають біометричні дані на локальній машині, можуть автентифікувати користувача тільки для роботи із цією машиною. Для великомасштабних інсталяцій і для поліпшення керованості рішень вибирайте системи із централізованим зберіганням. Якщо біометричне ПЗ розгорнуто на всіх вхідним у систему комп'ютерах, то користувачі, зареєструвавшись один раз, зможуть мати доступ до всіх ресурсів.

Для більшої надійності варто ввести реєстрацію кожного користувача по декількох біометричних характеристиках. Деякі пристрої дозволяють реєструвати, наприклад, відбитки всіх пальців на правій руці користувача. Якщо що-небудь трапилося з одним пальцем – поріз там або опік, то користувач вправі запропонувати для автентифікації інший палець, причому йому не прийдеться заново проходити реєстрацію.

У кожному разі вам доведеться використовувати апаратні й програмні засоби від одного постачальника – інтероперабельності в біометричної автентифікації дотепер не існує, незважаючи на старання консорціуму BioAPI Consortium виробити стандартні інтерфейси для інтеграції біометричних систем. Зате є додатки керування автентифікацією, подібні NMAS фірми Novell і SafeWord PremierAccess фірми Secure Computing, що інтегрують біометричні й небіометричні методи автентифікації для доступу до довідників.

Інтеграція додатків усе ще визначається взаєминами постачальників, тому дуже важливо переконатися, що обраний пристрій підтримує додатки або що постачальник не проти того, щоб зайнятися інтеграцією спеціально для вас. Інтеграція з настільними комп'ютерами або серверами звичайно здійснюється за допомогою модулів PAM (Plugable Authentication Module) для ОС Unix, GINA (Graphical Identification and Authentication) для ОС Windows або модулів Novell eDirectory LCM (Login Client Module). Увесь час, поки ім'я й пароль користувача зберігаються в кеш-пам'яті комп'ютера, вони можуть використовуватися для доступу до додатків. Однак якщо ваш додаток вимагає окремої реєстрації, то буває, що необхідної виявиться розробка додаткового ПЗ.

Для контролю доступу до критично важливим даних не слід застосовувати одні лише біометричних пристроїв, поки ви ретельно не протестуєте цю технологію. Якщо ваша мета полягає в тому, щоб забезпечити строгу автентифікацію, то задійте більше перевірені методи – апаратні й програмні жетони й паролі.

І, незважаючи на те, що ціни на біометричні пристрої останнім часом знижуються, старі, випробувані технології звичайно виявляються більше ефективними. Однак якщо вам подобається простота використання при більш-менш надійної автентифікації, те біометрична технологія, можливо, саме те, що потрібно.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки 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 першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) 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 кібербезпеки 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 кібербезпеки 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

Загалом, розпізнавання обличчя можна визначити як процес автоматичної ідентифікації особи шляхом аналізу зображення її обличчя. Цей процес охоплює цілий ряд проміжних завдань, від захоплення цифрового зображення до остаточного рішення щодо аналізованої особистості.

Щоб охопити всі етапи робочого процесу, необхідно використовувати різні технології, серед яких ключовими елементами є біометрія та машинне навчання:

– Біометрія: набір методів, які дозволяють за допомогою аналізу фізичних ознак (голосу, райдужної оболонки ока, відбитків пальців, малюнка вен, ДНК тощо) або поведінкових ознак (підпис, хода, взаємодія з цифровими інтерфейсами тощо) визначити особистість людини. Не всі біометричні ознаки однаково легко фіксуються і не забезпечують однакову ступінь впевненості, тому необхідно встановити компроміс між цими факторами, враховуючи вимоги кінцевої системи.

– Машинне навчання: галузь штучного інтелекту, яка прагне зробити машини здатними приймати рішення в ситуаціях, для яких вони явно не розроблені. Проблему біометричної перевірки особистості можна вирішити виключно з точки зору алгоритмів машинного навчання, оскільки завжди необхідно стикатися з новими прикладами (новими ідентифікаторами) на основі попереднього процесу навчання.

Методи алгебри та розробка алгоритмів машинного навчання

Сіровіч і Кірбі застосували методи лінійної алгебри для досягнення низьковимірних уявлень обличчя, тобто зведення найважливіших ознак обличчя до невеликого набору числових значень.

Ця робота була розширена Турком і Пентландом, щоб застосувати її до процесу виявлення обличчя, таким чином відкриваючи двері до повністю автоматичного процесу розпізнавання обличчя (без необхідності надавати попередньо обрізане зображення обличчя).

З цього моменту дослідницька діяльність була зосереджена на двох основних напрямках: на досягненні надійних дескрипторів обличчя та розробці алгоритмів машинного навчання, які дозволили б знайти шаблони та критерії для розрізнення цих характеристик, щоб розрізнити ідентичності.

Системи розпізнавання обличчя були успішно розгорнуті в ситуаціях, коли необхідно було автоматично підтвердити особу людини (наприклад, контроль доступу), а також служили альтернативою використанню традиційних паролів.

Тим не менш, нещодавній бум глибокого навчання змінив правила гри з точки зору значного підвищення продуктивності біометричних систем, що стало справжнім зміною парадигми та майже одностайним прийняттям галуззю.

З онтологічної точки зору, алгоритми глибокого навчання є підмножиною машинного навчання, що містить різні методи, які прагнуть отримати високорівневі абстракції шляхом аналізу складних зв’язків між великим набором вхідних даних. Основним елементом глибокого навчання є нейронні мережі, алгоритми з довгою історією, які досягли повної актуальності завдяки експоненціальному збільшенню обчислювальної потужності та доступності величезних обсягів даних.

Як зазначалося вище, з функціональної точки зору система розпізнавання облич містить кілька окремих етапів, які дозволяють успішно завершити процес:

1. Захоплення: початок робочого процесу полягає в отриманні основної інформації. У нашому випадку ця інформація є лише частиною обличчя вхідного зображення, тому першим автоматичним процесом буде виявлення обличчя.

Останніми роками була проведена інтенсивна робота в галузі алгоритмів розпізнавання облич, які наразі досягли дійсно високих показників ефективності

навіть у найскладніших умовах: штучне освітлення, дуже віддалені позиції, використання окулярів чи масок … Результат цього функціонального блоку буде кадрування області інтересу з вхідного зображення, щоб система з цього моменту могла працювати виключно з відповідною інформацією (область обличчя).

2. Обробка: у будь-якій системі, заснованій на машинному навчанні, обробка є фундаментальним етапом. Метою є нормалізація вхідних даних у межах попередньо встановлених параметрів, щоб продуктивність подальших алгоритмів була оптимальною.

Системи машинного навчання використовують висновки, зроблені в результаті аналізу навчального набору, щоб приймати рішення щодо нових прикладів у виробничих середовищах. З цієї причини вхідні дані повинні рухатися в межах параметрів, подібних до тих, що використовуються під час навчання, уникаючи крайніх випадків, які можуть призвести до небажаних результатів.

У випадку системи розпізнавання обличчя обробка складається з нормалізації вхідного зображення з точки зору числових значень (освітлення, відхилення кольору, діапазон значень кодування…), а також вмісту (центрування положення обличчя навколо вісь симетрії завдяки локалізації ключових точок обличчя).

3. Моделювання: це ключова операція в загальному процесі, оскільки вона перетворює вхідне зображення (вихід блоку обробки) у набір числових значень, широко відомих як вектор ознак. Вектор ознак можна розуміти як надійне кодування найважливіших аспектів обличчя, яке відрізняє його від інших облич.

Алгоритми штучного інтелекту аналізують складні шаблони у вхідних даних, щоб знайти ознаки з найвищою дискримінаційною здатністю та використовувати їх для моделювання вхідного зображення.

Наразі процес вилучення ознак здебільшого виконується алгоритмами глибокого навчання (зазвичай це згорточні мережеві архітектури), оскільки вони

здатні знаходити дуже складні (нелінійні) зв'язки у вхідних даних, значно перевершуючи ручні екстрактори ознак, які використовувалися в попередньому поколінні.

4. Порівняння: коли вектор ознак, пов'язаний із зображенням обличчя, отримано, можна розпочати процес порівняння. Мета полягає в тому, щоб визначити схожість (у числовому вираженні) між обличчями, щоб реалізувати одну з таких біометричних операцій: біометрична перевірка, біометрична ідентифікація або біометрична відповідність:

Перевірка: це я? Враховуючи вхідне зображення та ідентифікаційний тег, система повинна визначити, чи справді особа є тим, за кого себе видає, аналізуючи схожість між вхідним зображенням та іншим, попередньо збереженим у системі. Тому ця операція вимагає попередньої реєстрації.

Ідентифікація: хто я? За вхідного зображення система повинна визначити особу користувача шляхом порівняння з набором раніше зареєстрованих людей.

Відповідність: це та сама особа? Маючи два вхідних зображення, мета полягає в тому, щоб визначити відповідність між ними. Ця операція зазвичай використовується в системах віддаленої реєстрації клієнтів, де зображення документа, що посвідчує особу, порівнюється із зображенням, отриманим від користувача під час процесу реєстрації.

5. Прийняття рішень: нормалізоване числове значення, отримане на виході блоку порівняння, може бути не дуже описовим і тому потребує контекстуалізації. Мета етапу прийняття остаточного рішення полягає в тому, щоб повернути відповідь, яку можна інтерпретувати, з цього числового значення. Ця відповідь може бути двійковим значенням (ідентифікація підтверджена/непідтверджена) або мітка з ідентичністю, пов'язаною з введеним користувачем у разі ідентифікації.

Щоб прийняти ці рішення, необхідно встановити порогові значення для числових значень порівняння, щоб досягти балансу між безпекою (дуже суворі порогові значення, необхідний дуже високий ступінь впевненості в схожості) та

зручністю використання (більш м’які порогові значення, потрібен менший ступінь впевненості). Порогові значення для прийняття рішень необхідно адаптувати до потреб кожного проекту шляхом оцінки вартості більшої кількості помилок неправильної класифікації:

Помилковий результат: неправильне підтвердження особи людини. Це помилка, яка має високу вартість у програмах безпеки, де пропуск самозванця є серйозною загрозою.

Помилково негативний: неправильне відкидання особистості людини. Це помилка, яка має високу вартість у судово-медичних програмах, де неправильне відхилення особи людини може бути серйозною проблемою.

Опис алгоритму розпізнання обличчя

На першому кроці будується модель шкіри. Для цієї мети в кореневому каталозі системи повинна перебувати папка зі зразками шкіри (зображення 16x16). Зразки шкіри переводяться із простору RGB у простір YCbCr, і відкидається яскравісна складова Y. Потім обчислюються вибіркові середні для Cb і Cr складових і коваріаційна матриця, щоб можна було застосувати Гауссовську модель.

Після того, як була отримана модель шкіри, тестове зображення можна перевести в бінарне, тобто визначити для кожної ділянки шкіра це чи ні. А перш ніж зробити це необхідно обчислити для кожного пікселя величину, що відбиває ступінь подібності на шкіру. Потім величини подібності нормуються й зображення стає gray-scale, де чим світліше область, тим більше ймовірність того, що це шкіра.

Після того, як були розраховані величини подібності для кожного пікселя, на наступному кроці природно необхідно вибрати деякий поріг, і пікселі ступінь подібності яких на шкіру більше порога, будуть уважатися ділянками шкіри. Поріг може бути як фіксованим, так і адаптивним, у даній реалізації це адаптивний поріг. Після вибору найбільш оптимального порога, всі пікселі, значення яких вище порога стають рівними 1, а інші – 0.

Тепер, коли було отримано чорно-біле зображення, необхідно застосувати морфологічні операції, щоб відкинути такі ділянки, як коли одна особа перекриває інше або ділянки інших частин тіла. Порядок застосування морфологічних операцій має значення.

Спочатку застосовується морфологічне замикання до бінарного зображення.

Потім операція morphological erosion, за допомогою елемента диск розміру 10.

Після цього застосовується morphological dilation, щоб збільшити ті області бінарного зображення, які були зменшені на попередньому етапі.

Потім отримане зображення множиться на бінарне, щоб не втратилися всі світлі ділянки. Це необхідно для того, щоб на більше пізньому етапі за допомогою інформації про число чорних областей на білі можна було відфільтрувати деякі ділянки, які не є особами.

Кожну область на зображенні, що потенційно може бути особою, зафарбовуємо окремим кольором.

Метою цього кроку є відкинути ті білі ділянки, які не містять чорних областей, тому що на особі природно є не тільки чиста шкіра, але ще й губи, і ока

Перевірка пропорційності областей-претендентів, тут також ураховується, що особа з'єднана із шиєю, наявність вух.

Даний алгоритм показує гарні результати на базі ІММ, на тій базі, для якої програма й розроблялася. Крім того, функція локалізації особи може працювати на зображеннях різного розміру, з різним тлом і для зображень із людьми різних національностей.

Що стосується перспектив для даного алгоритму, те його можна доробити так, щоб на зображенні виділялися всі особи, які на ньому присутні. Така можливість уже є, але в силу того, що програма споконвічно припускала наявність однієї особи, вона поки не реалізована.

3.2 Розробка структурної схеми

Розглянемо розроблену структурну схему системи наведену на рисунку 3.1. Завдання розпізнавання (точніше, класифікації) об'єкта ставиться в такий спосіб. Є деякий спосіб кодування об'єктів (наприклад рукописних букв або зображення з мобільної камери), що належать заздалегідь відомій кінцевій множині класів $C=\{C_1,..., C_q\}$, і деяка кінцева множина об'єктів (навчальна множина), про кожний з яких відомо, якому класові він належить. Потрібно побудувати алгоритм, який по будь-якому вхідному об'єкту навчальній множині, що не обов'язково належить, розв'язує, якому класу цей об'єкт належить, і робить це достатньо добре. Якість розпізнавання оцінюється як імовірність (частота) помилки класифікації на іншій кінцевій множині об'єктів із заздалегідь відомими відповідями (тестовій множині).

При розпізнаванні на відміну від класифікації, буває потрібно оцінювати й чисельні характеристики об'єктів. Навчальна й тестова множина можуть не бути дані заздалегідь, а поповнюватися в процесі роботи алгоритму, що розпізнає. На вхід розпізнавання майже завжди потрапляють об'єкти, що не укладаються в класифікацію. Крім того бажані гарантії (звичайно, статистичні) того, що на будь-якій іншій тестовій множині частота помилки розпізнавання буде майже аналогічною.

Типова система розпізнавання складається із трьох частин:

– витяг ознак з картинки;

– власне розпізнавання;

– ухвалення рішення.

Витяг ознак – це перетворення вхідних об'єктів до однакового, компактного й зручного вигляду із втратою більшої частини інформації, що не впливає на класифікацію. Зручним виявляється представлення об'єкта точкою стандартного евклідового простору R, що належить деякому фіксованому полю

(кубу, кулі, сфері). Розмірність R повинна бути досить великою для успішного (якісного) розпізнавання.

Опис етапів алгоритму розпізнавання

На першому етапі алгоритм виділяє із загального зображення обличчя користувача смартфону. Потім відбувається нормалізація зображення. До нормалізації зображення ставляться наступні дії:

– Зміна роздрукової спроміжності зображення до 300 × 300 пікселів.

– Перетворення кольорів до 256 відтінків сірого.

– Зміна сумарної яскравості зображення до деякого середнього значення. Для деяких алгоритмів потрібно, щоб обличчя користувача смартфону на картинці розташовувався як можна більш вертикально.

У такому випадку поворот обличчя користувача смартфону на потрібний кут відбувається саме при нормалізації. Наступним етапом алгоритму розпізнавання є виділення характеристик наявного обличчя користувача смартфону. Виділювані характеристики сильно залежать від алгоритму розпізнавання, тому їх приклади будуть наведені пізніше. Зауважимо, що після виділення характеристик картинка більше не потрібна.

Останнім етапом розпізнавання є застосування класифікатора, який по наявних характеристиках видає відповідь на завдання – обличчя користувача смартфону співпадають чи ні. І як наслідок користувач одержує доступ до ПК.

Опис уніфікації зображення з обличчям користувача смартфону

На всіх картинках з обличчям користувача смартфону співвідношення між точками приблизно однакові. Тому, доцільно виділити на обличчі користувача смартфону деякі контрольні точки. Після цього для ідентифікації того або іншого обличчя користувача смартфону, досить зчитати значення фільтрів Габора саме в контрольних точках.

Виділимо на довільному обличчі користувача смартфону деякі контрольні точки. Для того, щоб визначити, де перебувають ці точки, необхідно мати деякі

представлення про їхнє розташування. Для цього уніфікуємо зображення з обличчям користувача смартфону.

Зафіксуємо набір контрольних точок, що цікавлять нас. Потім для кожного зображення з колекції вкажемо, де знаходиться кожна з контрольних точок, і порахуємо значення 40 фільтрів Габора в цих точках. Вектор, що складається з 40 значень фільтрів Габора, полічених у конкретній точці називається jet'ом цієї точки.

Тепер візьмемо середні значення відстані між контрольними точками. Також візьмемо середні значення jet'ів. Разом, ми одержали деякий граф, у якому вершинам відповідають контрольні точки, а довжини ребер дорівнюють середнім відстаням між даними контрольними точками.

Крім того, у кожній вершині зберігається один середній jet. Отриманий у такий спосіб граф називається уніфіковане зображення з обличчям користувача смартфону.

Опис знаходження контрольних точок

Після одержання на вхід нового обличчя користувача смартфону насамперед треба знайти положення контрольних точок на ньому. Для кожної контрольної точки відомий її jet.

Необхідно знайти такий вектор точок, щоб jet кожної точки вектора був якнайближче до jet'у відповідної контрольної точки. При цьому, також щоб відстані між обраними крапками були як можна більш пропорційні довжинам ребер уніфікуємого графа. Є досить багато методів мінімізації різних функцій. Для даного випадку підходить дуже простий метод. Спочатку великими кроками паралельно переміщаємо решітку та уніфікуємо зображення й порівнюємо, що виходять jet'и контрольних точок з еталонними.

Далі після того, як з'ясували приблизне розташування решітки, робимо зсув й повторюємо дію.

Потім незалежно переміщаємо кожну контрольну точку на незначну відстань. Цей метод використовує не повністю мінімизуючий набір точок, але в

задачі розпізнавання обличчі користувача смартфону в ці точки можна вважати контрольними точками зображення.

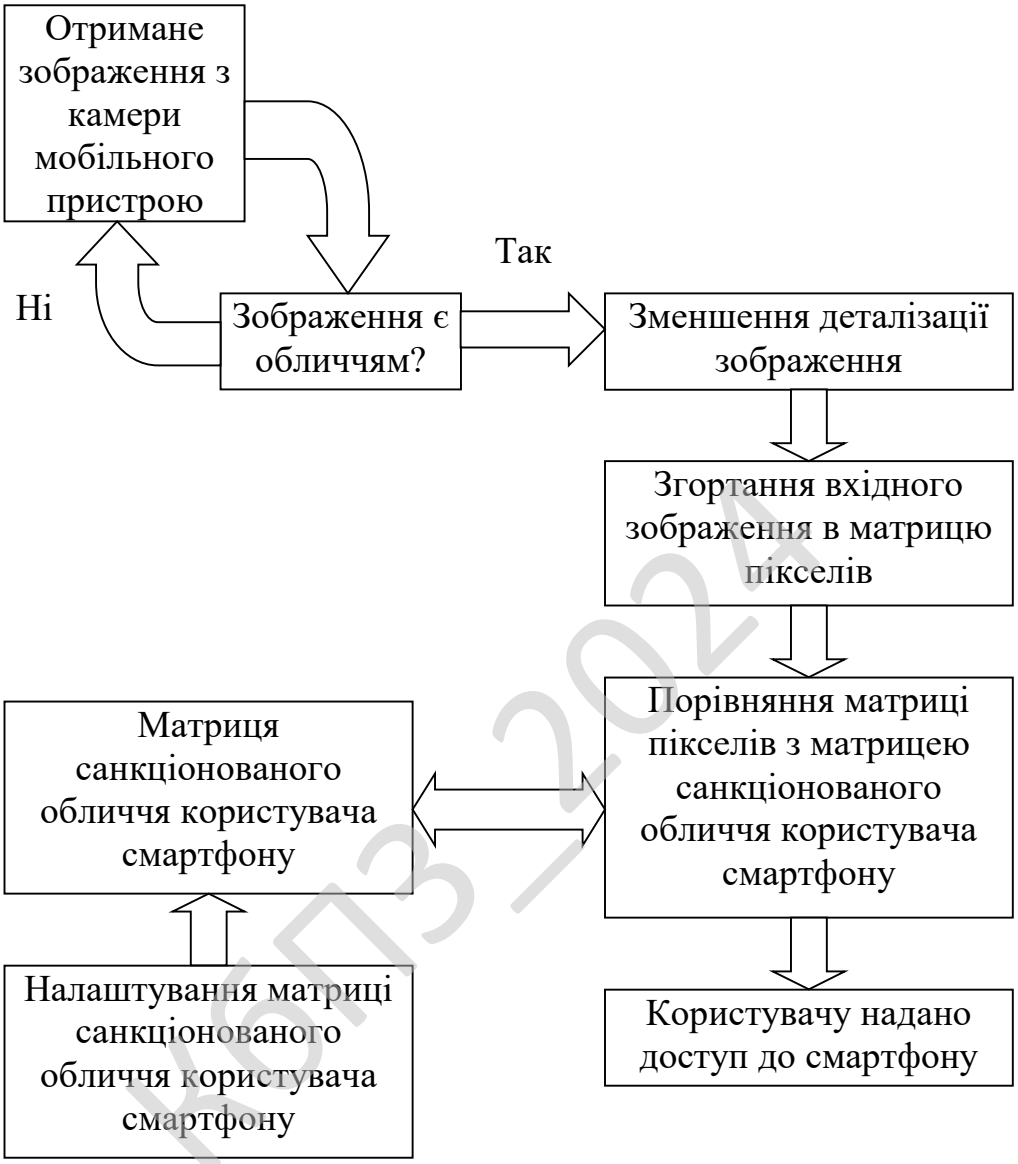

Рисунок 3.1 – Структурна схема системи

Опис і обґрунтування класифікації

Після того, як знайдені контрольні точки зображення, ми маємо вектор нового зображення, що є набором jet'ів усіх контрольних точках. Потрібно визначити, якому обличчя користувача смартфону відповідає даний вектор. Для цього застосовуються стандартні методи класифікації. Переваги – метод

класифікації графів має поруч перевагу навчання, для досягнення гарних результатів йому необхідна маленька вихідна колекція облич користувача смартфону.

Структурна схема зображує систему біометричної ідентифікації обличчя користувача смартфону. Спочатку проходе отримання зображення з камери мобільного пристрою потім якщо це зображення є обличчям користувача смартфону проходить зменшення деталізації зображення для порівняння.

Зображення отримується шляхом фіксації кадру з роздрукованим обличчям користувача смартфону на листку папера. Якщо згортання пройшло невдало проходе ще спроба доти поки вхідне отримане зображення не буде згорнуто до розмірів матриці пікселів.

Проводиться порівняння матриці пікселів з матрицею обличчя користувача смартфону санкціонованого доступу через існуючу матрицю обличчя користувача смартфону санкціонованого доступу.

На зображенні виділяються характеристики та проходить розпізнавання зображення з застосуванням класифікатора (вибір із шаблонів шрифтів). Якщо зображення є обличчям користувача смартфону проходить ідентифікація обличчя користувача смартфону та встановлюється обличчя користувача смартфону ідентифіковано чи ні. І якщо обличчя користувача смартфону ідентифіковано розроблене ПЗ дає доступ до мобільно пристрою.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Вхідні дані це отримання обличчя користувача смартфону доступу до мобільного пристрою через камеру мобільного пристрою.

У мобільному пристрої через модуль роботи з камерою мобільного пристрою проходить зменшення деталізації зображення, проходить підгін до формату *.bmp. Проходить його обробка та в кінцевому випадку на виході ми

отримуємо доступ.

Розроблене ПЗ складається з наступних функціональних блоків: Інтерфейс ПЗ; Конвертація інформації у бітовий потік; Аналіз графічного зображення; Сегментація зображення; Порівняння матриці пікселів; Аналіз ознак зображення обличчя користувача смартфону; Синтаксичний контроль; Налаштування ПЗ; Редагування матриці обличчя користувача смартфону санкціонованого доступу.

При рішенні завдання розпізнавання картинок виникають дві проблеми. По-перше, будь-яка картинка являє собою масив пікселів.

У той же час один піксель картинки нічого не значить (його колір можна змінити, і ніхто не помітить різниці). Це робить таке представлення картинок надлишковим й неекономічним.

Таким чином, для ефективного розпізнавання картинок необхідно розробити деякий компактний і зручний формат представлення картинок.

Задача яка розглядається у дипломі це задача розпізнавання образів яка є однією з найбільш фундаментальних проблем теорії інтелектуальних систем. З іншого боку, задача розпізнавання образів має величезне практичне значення. Замість терміну "розпізнавання" часто використовується інший термін – "класифікація". Ці два терміни у багатьох випадках розглядаються як синоніми, але не є повністю взаємно замінюваними.

Кожний з цих термінів має свої сфери застосування, і інтерпретація обох термінів часто залежить від специфіки конкретної задачі.

Основна проблема полягає у тому, що часто неможливо адекватно визначити ознаки, на основі яких слід здійснювати розпізнавання.

Для задач, для яких такі ознаки вдається виділити, штучні системи розпізнавання набули значного поширення і широко використовуються.

Рисунок 3.2 – Функціональна схема роботи системи

Методи розпізнавання образів та технічні системи, що реалізують ці методи, широко використовуються на практиці. На сьогоднішній день відома безліч способів стиску зображень із втратами, але використовуваний у ньому формат не зручний.

Це зв'язано в першу чергу з тим, що немає необхідності визначати, як виглядає дана картинка з колекції, а потрібно розв'язати зворотне завдання: яка картинка з колекції виглядає даним образом.

Друга проблема полягає в тому, що одна й те ж картинка може бути роздрукована при різних зовнішніх факторах. Окрім усього іншого, ця проблема має велике значення для власне комп'ютерних технологій.

Системи розпізнавання літер працюють разом зі сканерами – пристроями, які використовуються для введення до комп'ютера друкованих зображень і текстів.

При введенні друкованого тексту сканер формує лише графічне зображення; для того, щоб створити текстовий документ, з яким може працювати текстовий редактор, необхідно впізнати на цьому зображенні окремі літери.

Становлення біометрії йде рука об руку зі стандартизацією її основних функцій, форматів і процесів. Стандартизація підвищує функціональність технології, спрощує її впровадження й інтеграцію, скорочує час розробки систем на її основі. Біометрія історично є технологією, яка характеризується надлишковою складністю розробки й широким спектром як методів біометричного розпізнавання, так і областей їхнього застосування. Розробка й введення стандартів для всіх рівнів життєвого циклу біометрії, від формування шаблону біометричної характеристики до правил використання біометричних систем у певній області, заслуговують на пильну увагу.

Розглянемо стандарти, зазначені в ієрархії, і організації, що займаються їхнім створенням. Виключення зробимо лише для стандартів за розрахунками продуктивності й інших технічних характеристик біометричних систем, тому що вони зараз самі "сирі" і багато в чому прямо залежать від стану й твердження стандартів інших груп.

Група M1 і підкомітет SC37

У США при Міжнародному комітеті зі стандартам в інформаційних технологіях (International Committee for IT Standards, INCITS) був створений технічний комітет M1, основним завданням якого стала прискорена розробка стандартів по біометрії для використання в США й у міжнародних стандартах.

На міжнародному рівні цими завданнями займається підкомітет SC37 (Subcommittee 37) об'єднаного технічного комітету з інформаційним технологіям JTC 1 Міжнародної організації стандартизації ISO (International Organization for Standardization) і Міжнародної електротехнічної комісії (International Electrotechnical Commission, IEC. У нього входить близько 20 країн. У нашій країні створений підкомітет N 7 по біометрії в рамках Технічного комітету Держстандарту ТК355 по автоматичній ідентифікації, що діє в складі SC37 і

займається адаптацією й випуском російських стандартів по біометрії. Група М1 входить в SC37 як представник США й технічна консультаційна група (Technical Advisory Group).

Напрямки діяльності М1 і SC37 аналогічні, тому зупинимося на описі робіт М1, тому що вони почалися раніше й деякі з них лягли в основу біометричних стандартів JTC1 SC 37 ISO/IEC.

Усередині М1 виділено чотири групи, кожна з яких відповідає за розробку своєї частини стандартів.

Група М1.1 займається стандартами обміну біометричними даними для чотирьох найпоширеніших технологій розпізнавання. У стадії розробки перебувають наступні стандарти:

− формат шаблону відбитка пальця, сформованого по точках;

− формат шаблону відбитка пальця, сформованого по візерунку;

− формат зображення відбитка пальця;

− формат шаблону для розпізнавання за особою;

− формат шаблону для розпізнавання по райдужній оболонці ока;

− формат шаблону для розпізнавання по підпису й рукописних даних;

− формат шаблону для розпізнавання по геометрії руки.

Група М1.2 розробляє стандарти, що визначають вимоги до побудови захищеної взаємодії між біометричними компонентами й різними системами.

Група М1.3 трудиться над стандартами, орієнтованими на використання біометрії в конкретних промислових областях. У цей час у розробці перебувають три стандарти, що стосуються застосування:

– біометричній ідентифікації й верифікації в транспортній промисловості (Application Profile Verification & Identification of Transportation Workers);

– біометричних систем у цивільній ідентифікації при перетинанні границь держав (Application Profile Personal Identification for Border Crossing);

– біометричної верифікації в магазинах (Application Profile Biometric Verification in Point-of-Sale Systems).

Група М1.4 займається стандартизацією методів тестування продуктивності, точності й інших технічних характеристик біометричних систем.

Слід зазначити, що всі стандарти комітету М1 перебувають зараз у стадії розробки.

Стандарти, що визначають прикладний програмний інтерфейс для розробки біометричних систем

Дана група представлена по суті всього двома стандартами: BioAPI і BAPI (їх було більше, але згодом всі вони в тім або іншому ступені ввійшли в BioAPI). Специфікації BioAPI V1.1 розроблені Міжнародним біометричним консорціумом BioAPI, у складі якого більше 80 організацій і компаній-виробників.

Специфікації BioAPI версії 1.1. затверджені Американським національним інститутом стандартів (ANSI) і INCITS як стандарт ANSI/INCITS 358-2002, що визначає прикладний програмний інтерфейс (API) і інтерфейс провайдеру послуг (Service Provider Interface, SPA) для біометричних технологій.

Поява єдиного індустріального біометричного стандарту стало результатом об'єднання вже існуючих. Основною метою консорціуму BioAPI було створення API:

– незалежного від реалізованого методу біометричного розпізнавання;

– незалежного від операційної системи;

– застосовного в будь-яких областях, де для рішення поставлених завдань потрібна біометрія.

В інтерфейсі BioAPI виділено два рівні: верхній, визначальний інтерфейс клієнтського й серверного додатків, що викликає функції біометричного розпізнавання й реєстрації, і нижній, визначальний інтерфейс взаємодії із провайдером біометричних послуг (Biometric Service Provider, BSP), що виконують виклики верхнього рівня.

Верхній рівень визначає три основні функції, необхідні додатку для проведення біометричної автентифікації:

– Enroll (реєстрація). Виміру з біометричного пристрою, що зчитує, трансформується в придатний для використання вид, з якого формується шаблон, переданий додатку;

– Verify (верифікація, порівняння "один до одному"). Одне або більше вимірів знімається з біометричного пристрою, переводиться в придатну для використання форму й рівняється з відповідним шаблоном. Результати передаються додатку;

– Identify (ідентифікація, порівняння "один до багатьох"). Одне або кілька вимірів знімається з біометричного пристрою, переводиться в придатну для використання форму й рівняється з набором шаблонів. У результаті передається список, що показує, наскільки близько виміру збігаються з кандидатами на ідентифікацію з набору шаблонів.

Нижній рівень, SPI, визначає інтерфейс BSP, у якості якого можуть виступати практично будь-які сумісні із цим інтерфейсом біометричні системи, пристрої або програмні продукти. Функція SPI – це відображення "один до одному" викликів верхнього рівня у виклики до BSP.

Така архітектура BioAPI визначила поділ компаній – учасників біометричного ринку на дві категорії:

– розроблювачі рішень верхнього рівня, тобто вбудовуються в різні додатки (наприклад, для заміни парольного захисту в якій-небудь програмній системі). Для позначення таких продуктів-посередників між кінцевим устаткуванням і системами, у які вбудовується біометрія, використовується термін Middleware (проміжне програмне забезпечення);

– розроблювачі біометричного встаткування й інтерфейсу BSP для роботи з ним. В BSP реалізується безпосередньо алгоритм біометричного розпізнавання й роботи зі скануючим пристроєм.

Таким чином, кінцевий споживач технології одержує в деякому змісті програмно-апаратну незалежність при роботі з різними біометричними системами. Тобто в тому випадку, коли його влаштовує використовуване їм

middleware, але не влаштовує біометричне встаткування, він просто міняє його на інше BioAPI-сумісне встаткування (з відповідної BSP). При необхідності зміни встаткування й технології розпізнавання споживач просто заново набирає базу біометричних ідентифікаторів для зареєстрованих користувачів. Якщо справа є навпаки, то залишається його встаткування, що влаштовує, і здобувається інше middleware.

Обробка біометричних даних з моменту одержання вимірів із пристрою й до моменту порівняння їх із шаблоном може розділятися на безліч етапів з різним обчислювальним навантаженням. Виходячи із цього інтерфейс BioAPI надає розроблювачам широкі можливості по розподілі обчислювального навантаження між клієнтським місцем (яке має біометричний пристрій, що зчитує) і сервером, а також дозволяє повністю зосередити її тільки в автономному біометричному пристрої.

Основна одиниця подання даних в BioAPI – біометричний ідентифікаційний запис BIR (Biometric Identification Record); це набір біометричної інформації, з якою працюють додатки верхнього рівня й провайдери послуг. Формат BIR визначається стандартом CBEFF, мова про яке піде нижче.

Крім того, в BioAPI передбачені механізми визначення фактичних значень основних імовірнісних показників якості біометричного розпізнавання:

– FAR (false accept rate) – імовірність ухвалення помилкового рішення про ідентичність порівнюваних шаблонів, тобто ймовірність "пропустити чужого");

– FRR (false reject rate) – імовірність ухвалення помилкового рішення про неідентичність однакових шаблонів, тобто ймовірність "не пропустити свого").

Визначення фактичних значень FAR і FRR – дуже корисна можливість, оскільки дозволяє перевірити відповідність імовірнісних характеристик біометричної системи, що заявляються, реальним характеристикам для конкретного користувача.

У цілому в основу BioAPI закладені ті ж принципи, що й в CryptoAPI, тільки з урахуванням специфіки біометричних технологій. Схожість підходу полягає в тім, що в BioAPI прикладна програма викликає функції тільки верхнього рівня й ніяк не прив'язана до прикінцевого встаткування, так само як і CryptoAPI не прив'язаний до криптомодулів.

Практично всі наробітки різних компаній і організацій по створенню API для біометричних технологій об'єднані й враховані при створенні BioAPI. Окремий розвиток одержав лише стандарт BAPI (Biometric Application Programming Interface) компанії I/O Software. У деякому змісті його можна вважати адаптацією BioAPI для використання в операційних системах сімейства MS Windows. Незважаючи на це BAPI є самостійним стандартом з комплекту стандартів по захисту інформації I/O Software, орієнтованим на використання біометричного встаткування різних виробників у захищених системах цієї компанії.

Варто виділити ще три адаптації BioAPI під різні операційні системи: Міжнародна біометрична група (International Biometric Group, IBG) працює над BioAPI під Solaris, NIST – над BioAPI під Linux і спільно NIST і IBG – над BioAPI під UNIX. Ці розробки вже завершені й зараз перебувають у стадії тестування.

Єдиний формат подання біометричних даних

Базовим у даній групі є стандарт Common Biometric Exchange File Format (CBEFF), розроблений NIST при сприянні біометричного консорціуму NISTIR 6529.

CBEFF – це єдиний формат подання біометричних даних, запропонований для заміни різних біометричних форматів, використовуваних виробниками біометричного встаткування й ПЗ. При створенні CBEFF були враховані всі можливі аспекти його застосування, у тому числі криптографія, багатофакторна біометрична ідентифікація й інтеграція з картковими системами ідентифікації.

Пакет біометричних даних у форматі CBEFF складається із трьох секцій даних: заголовка, біометричних даних і електронний цифрового підпису (ЕЦП).

Дамо пояснення по кожному з полів заголовка й двох секцій, що залишилися, CBEFF, тому що ці дані дозволять краще зрозуміти універсальність використання цього формату для всіляких технологій і додатків.

Ступінь захисту біометричних даних – указує тип захисту біометричних даних, може приймати одне із чотирьох значень: немає захисту, зашифровані, використовується ЕЦП або імітовставка, зашифровані й підписані ЕЦП.

Контроль цілісності біометричних даних – указує ступінь цілісності даних, можливі наступні варіанти: ні, ЕЦП або імітовставка.

Версія заголовка CBEFF – номер версії заголовка визначає обсяг даних, що вказуються.

Обличчя користувача смартфону реалізації заголовка – визначає, яка з реалізацій заголовка CBEFF використовується. Реалізація заголовка – це його спеціалізація (патронний формат) для застосування з різними стандартами (можуть бути включені поля, не задіяні в CBEFF, а опціональні поля CBEFF використовуватися по-іншому).

Можливі дві реалізації CBEFF: під стандарти BioAPI – BIR і X9.84 – Biometric Object. Але в стандарті передбачені варіанти розширення й під інші реалізації CBEFF, а у відновленні стандарту планується описати версії під стандарти AAMVA, XCBF, ISO SC17 7816-11.

Тип біометричних даних – визначає застосовувану біометричну технологію. Зареєстровано 19 методів розпізнавання (можливе розширення даного списку при твердженні наступної версії формату): комбінація методів, методи, засновані на розпізнаванні форми особи, голоси, відбитків пальців, сітківки ока, райдужної оболонки, геометрії кисті руки, динаміки підпису, динаміки клавіатурного набору, руху губ, термографії особи, термографії руки, ходи, заходу тіла, ДНК, форми вуха, геометрії пальця, геометрії долоні й малюнка вен на руці.

Стан даних – це поле показує стадію обробки біометричних даних у різних системах, на якій перебувають передані дані: неопрацьовані, передоброблені, оброблені.

Якість даних – для деяких методів біометричного розпізнавання необхідно вказувати якість переданих даних, вона оцінюється числами від 0 до 100. Якщо зазначено -1, то якість не задана, якщо -2, то в даному біометричному методі таке поняття, як "якість", не використовується.

Дата створення – містить час і дату одержання біометричних даних.

Власник формату подання біометричних даних – у цьому полі вказується обличчя користувача смартфону компанії-власника формату. Всі формати подання безпосередньо біометричних даних (тобто вимірів зі зчитувача), сумісних з CBEFF, реєструються в Міжнародній промисловій біометричній асоціації IBIA (International Biometric Industry Association). Сьогодні там зафіксовано 27 форматів різних компаній, серед яких такі великі виробники, як SAFLINK, Bioscrypt, Identix, Infineon, Iridian Technologies, Veridicom, Cyber SIGN, Fingerprint Cards, SecuGen, Precise Biometric.

Тип формату подання даних – номер формату із зареєстрованих власником (наприклад, компанія LOGICO Smartcard Solutions має два формати подання даних).

Блок біометричних даних – містить безпосередньо біометричні дані (у відкритому або зашифрованому виді), використовувані для розпізнавання людини відповідно до зареєстрованого формату, зазначеним у заголовку.

Блок ЕЦП – необов'язкове поле, у нього заноситься електронний цифровий підпис. Якщо ЕЦП використовується, то підписуються разом і заголовок, і біометричні дані.

Спеціалізовані стандарти

На закінчення розглянемо кілька спеціалізованих стандартів, у докладному описі яких немає особою необхідності через їхню специфічність, але

сказати про неї коротенько має сенс, тому що вони дозволяють продемонструвати всю широту спектра стандартизації біометрії.

XCBF (XML Common Biometric Format) – стандарт, розроблений технічним комітетом OASIS. Він визначає набір криптографічних повідомлень, представлених у вигляді XML-тегів, які можуть бути використані для безпечного збору, обробки й зберігання біометричної інформації. Сполучимо зі специфікаціями BioAPI і стандартами X9.84 і CBEFF.

AAMVA Fingerprint Minutiae Format/National Standard for the Driver License/Identification Card DL/ ID-2000 – американський стандарт на формат подання, зберігання й передачі відбитків пальців для прав водія. Сполучимо зі специфікаціями BioAPI і стандартом CBEFF.

CDSA/HRS (Human Recognition Services) – являє собою біометричний модуль в архітектурі CommonData Security Architecture, розроблений Intel Architecture Labs. Він схвалений консорціумом Open Group і визначає набір API-інтерфейсів, що представляють собою логічно зв'язану безліч функцій, що охоплюють такі компоненти захисту, як шифрування, цифрові сертифікати, різні способи автентифікації користувачів, у список яких за допомогою HRS додана й біометрія. CDSA/HRS сполучимо зі специфікаціями BioAPI і стандартом CBEFF.

ANSI/ NIST-ITL 1-2000 Fingerprint Standard Revision – американський стандарт, що визначає загальний формат подання й передачі даних по відбитках пальців, особі, натільним шрамам і татуюванням для використання в правоохоронних органах США.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.3.

Після початку роботи розробленого ПЗ ми потрапляємо до інтерфейсу ПЗ звідки можемо потрапити до налаштувань ПЗ, бібліотеки швидкого сканування відеопотоку та через підключення камери та отримання зображення потрапити до згортання вхідного зображення в матрицю. Далі ми можемо потрапити до зменшення деталізації зображення та до матриці коду санкціонованого доступу. З матриці коду санкціонованого доступу до налаштування матриці коду санкціонованого доступу та далі з матриці до порівняння матриці з матрицею коду та остаточному санкціонуванню доступу.

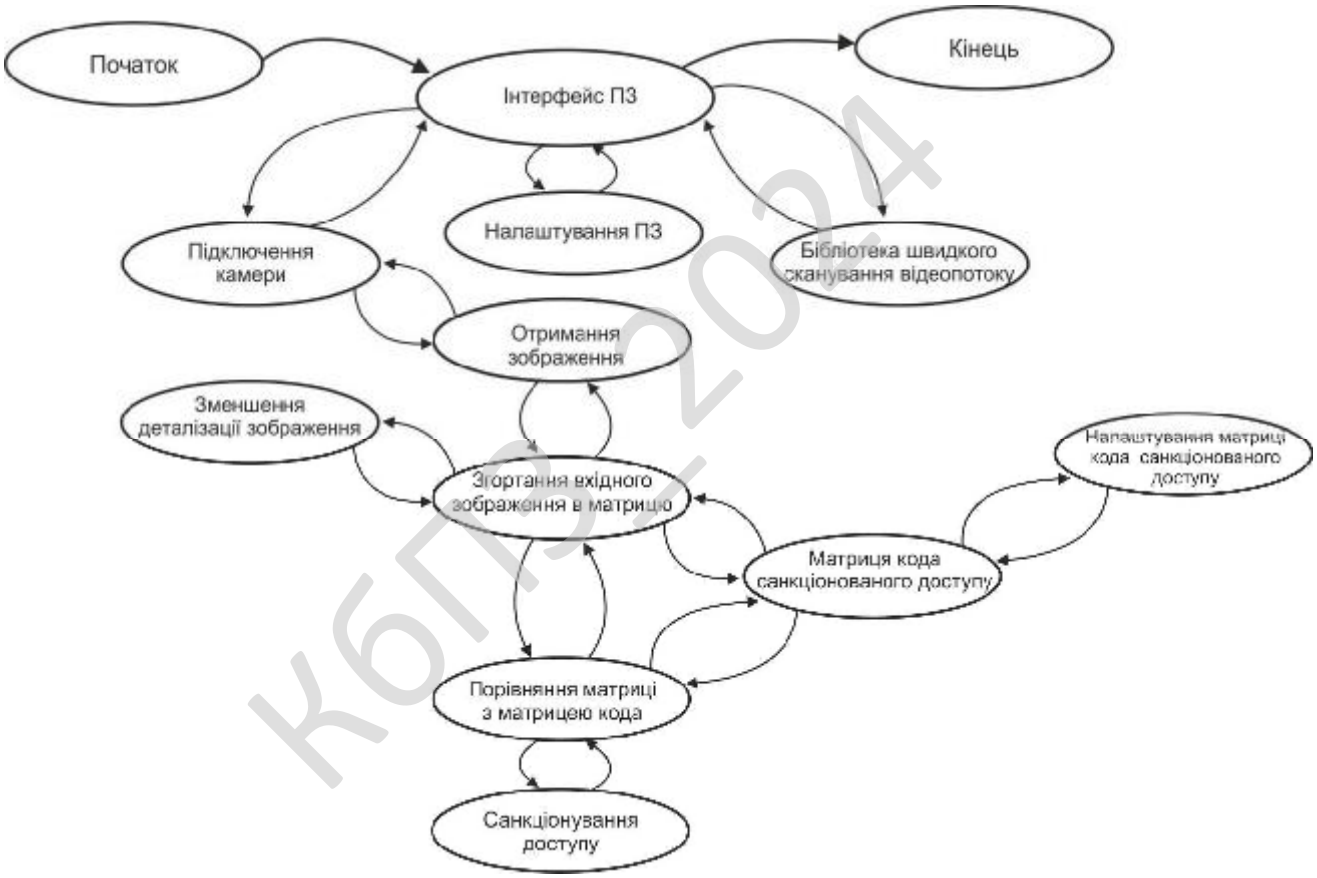

Рисунок 3.3 – Діаграма взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків:

– Початкова ініціалізація ПЗ.

– Завантаження файлу налаштувань фокусування.

– Завантаження файлу прав доступу.

– Завантаження файлу матриці коду.

– Файли завантажено?

– Підключення до модуля камери смартфону.

– Модуль підключено?

– Виведення вікна пошуку обличчя користувача.

– Очікування суттєвого руху у фокусі мобільної камери.

– Зображення змінено?

– Підпрограма пошуку обличчя користувача (рисунок 4.2).

– Формування зображення обличчя користувача.

– Виведення зображення обличчя користувача.

– Перевірка прав доступу користувача.

– Доступ отримано?

– Приховання вікна пошуку обличчя користувача.

– Виведення вікна отримання доступу до ПК.

– сигнал aTerminate?

– Звільнення виділених динамічних ресурсів користувача.

– Звільнення призначених для користувача типів даних.

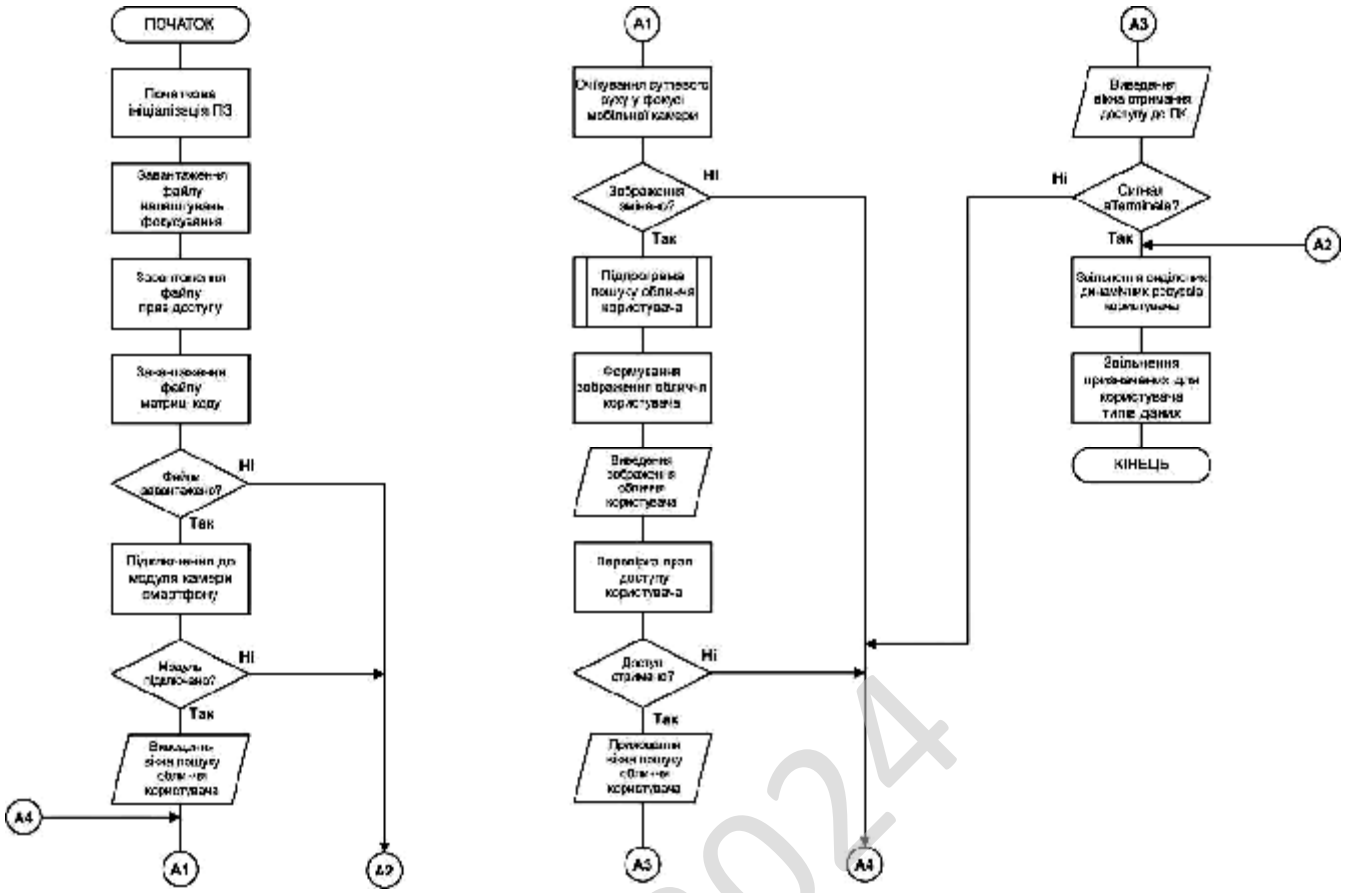

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 наведено блок-схему підпрограми пошуку обличчя користувача. Її робота складається з виконання наступних кроків:

– Отримання кадру з камери смарфону.

– Швидка конвертація форматів Jpg у Bmp.

– Переведення у палітру 256 відтінків сірого.

– Зменшення розміру зображення.

– Сканування та створення динамічного масиву зображення.

– Зчитування рядка пікселів у динамічний масив.

– Пошук та видалення полутонів у рядку.

– Додавання рядку до матриці зображення.

– Всі рядки оброблені?

– Читання файлу зразків.

– Формування масиву користувачів (обличчя).

– Заповнення матриці обличчя.

– Застосування класифікатору.

– Виділення області розпізнавання обличчя.

– Порівняння області обличчя з масивом користувачів.

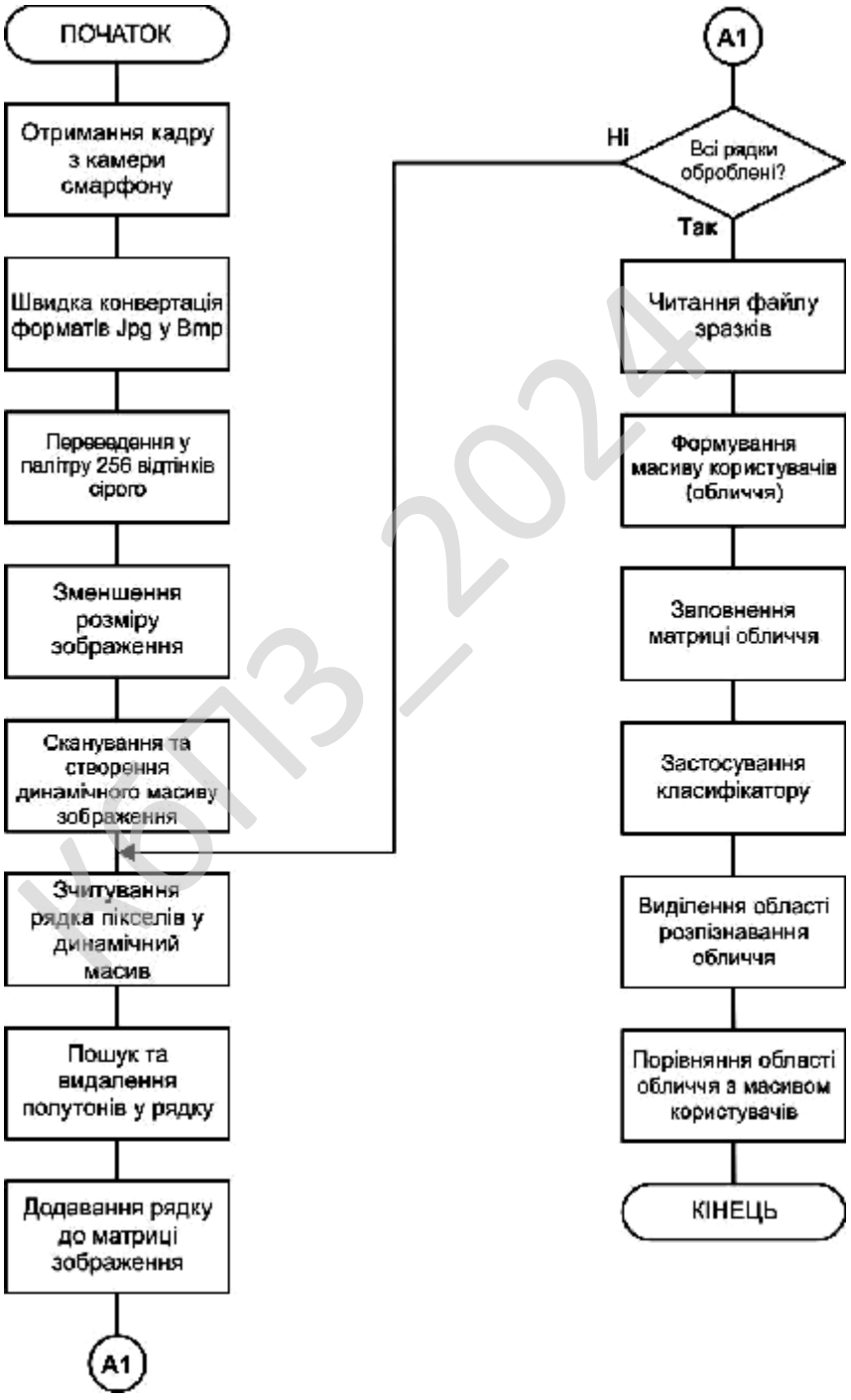

Рисунок 4.2 – Блок-схема підпрограми пошуку обличчя користувача

Система розпізнавання осіб по зображенню ґрунтується на алгоритмах ідентифікації та порівняння зображень.

Базою для цих алгоритмів є модифікований метод аналізу принципових компонент, який полягає в обчисленні максимально схожих коефіцієнтів, які характеризують вхідні образи розпізнавання осіб людини.

Інноваційна система розпізнавання осіб дозволяє забезпечити високу ймовірність розпізнавання осіб, в тому числі, при зміні фізичних характеристик особи: старінні, зміні зачіски, появі бороди і вусів. Розпізнавання може проводитися при знаходженні людини на відстані до двох метрів.

Розглянемо опис алгоритмів функціонування системи

Алгоритм розпізнавання лиць

Системи розпізнавання лиць діляться дві категорії – двомірні (в їх основі лежать плоскі, або двомірні, зображення, 2D) і тривимірні (розпізнавання проводиться за реконструйованим тривимірним образам, 3D).

Однак, системи 2D – розпізнавання дуже чутливі до умов освітленості. При нерівномірному освітленні особи достовірність 2D – розпізнавання помітно знижується.

У той час як для систем 3D – розпізнавання зміни в освітленості впливають лише на текстуру особи, а реконструйована поверхню обличчя залишається незмінною. І в тих і в інших системах для розпізнавання лиць застосовуються стійкі антропометричні точки, розташування яких характеризує індивідуальні особливості особи.

На 3D-моделях антропометричні точки визначаються з більшою точністю, ніж на 2D-зображеннях. Також, точки на 3D-моделях мають три координати і, тому, надають більше інформації, ніж ті такі ж точки на 2D-зображенні.

Розпізнавання проводиться шляхом виділення симетрій в кожному відео зображенні рисунок 4.3. Для цього використовується певний набір симетричних згорток у заданому діапазоні масштабів зображення, після цього відеокадр обробляється нейромережею.

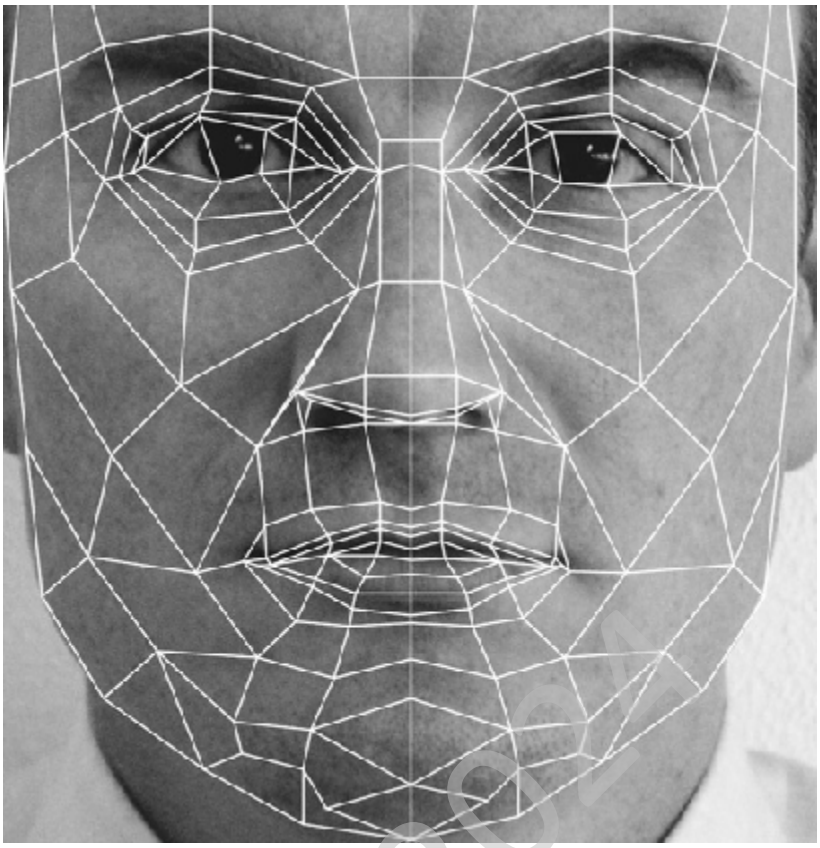

Рисунок 4.3 – Виділення симетрій в відеозображенні

Алгоритм розпізнавання лиць дозволяє забезпечити стійкість до шуму і нерівномірного засвітке. Обличчя людини, хоч раз потрапило в поле зору відеокамери, із застосуванням алгоритму передбачення вектора руху буде автоматично відслідковуватися від кадру до кадру.

А всі зображення будуть зберігатися в тимчасовому буфері. У результаті буде відібраний кадр з оптимальним ракурсом особи і якістю зображення. Потім буде зроблено виділення основних ознак особи: очі, ніс, рот. Після виявлення основних ознак особи, зображення приводиться до стандартного вигляду: для надійного розпізнавання зображення обличчя повинно мати певні розміри, необхідно витримати відстань між очима, становище особи щодо центру.

Для цього зображення масштабується, розгортається, в деяких випадках визначається становище особи (фас, положення в три чверті або точні 3D координати), автоматично нормалізується контрастність і яскравість.

Після цього здійснюється безпосереднє порівняння отриманого зображення особи з зображеннями з бази даних, яке зберігається в такому ж форматі.

У підсумку порівняння "один до багатьох " відбираються найбільш близькі за характеристиками вектора: результатом заключного етапу є ідентифікація особи, яке потрапило в поле зору відеокамери з зображеннями з бази даних.

Функціональні можливості і характеристики дозволяють успішно використовувати систему розпізнавання лиць на самих різних об'єктах, для забезпечення безпеки яких важлива обов'язкова реєстрація та надання даних про людей, які знаходяться на території об'єкта, а також їх ідентифікація.

Алгоритм розпізнавання людських облич

ПЗ призначено для розпізнавання людських облич на фотографіях відео потоку. Система працює в 2 етапи.

1. Це процес навчання під час якого в систему (в блок попередньої обробки) подаються зображення і класи (як класу виступає конкретна людина) яким відповідають ці зображення. Налаштування системи захисту – завантаження користувачів смартфону.

2. безпосередня класифікація, яка полягає в зіставленні вхідного зображення певного класу. Під час цього етапу в систему подається відеопотік. Вхідні дані обробляються і результати подаються в блок розпізнавання. У блоці розпізнавання вхідні дані, які характеризують зображення, зіставляються певних класів, яким вони відповідають.

Програма на етапі розпізнавання на вхід може приймати як фотографії, так і відеопотік з web – камери або відеофайлу.

Система складається з 4 -х блоків:

– Блок вилучення зображення.

– Блок попередньої обробки.

– Блок перетворення і виділення ключових ознак.

– Блок розпізнавання.

Блок вилучення зображення

У блоці вилучення зображення відбувається операція витягання осіб з вхідного зображення. Для цього завдання використана бібліотека комп'ютерного зору OpenCv.

Вихідний код для сегментації:

```
// Захоплюємо зображення з пристрою введення
pFrame = cvQueryFrame ( pCapture ) ;
// Перехід в GrayScale
cvCvtColor ( pFrame, pFrameGray, CV_BGR2GRAY ) ;
// Попереднє зменшення зображення для прискорення роботи класифікатора
if ( scale ! = 1 )
{
    cvResize ( pFrameGray, pSmallFrame, CV_INTER_LINEAR ) ;
    pRecFrame = pSmallFrame ;
}
else
    pRecFrame = pFrameGray ;
// Виділення областей осіб на зображенні
// PRecFrame – вихідне зображення
// ClassFace – навчений класифікатор
//  ( поставляються з бібліотекою у вигляді xml файлів)
// PStorage – ініціалізованих область пам'яті
// необхідна для роботи функції
//
CvSeq * pSeq = cvHaarDetectObjects ( pRecFrame, classFace, pStorage,
                          1.2, 2, 0 | CV_HAAR_DO_CANNY_PRUNING,
                          cvSize ( 30, 30 )) ;

CvvImage imgFace ;
for ( int i = 0 ; i < pSeq – > total ; i + +)
{
    CvRect r = ( CvRect * ) cvGetSeqElem ( pSeq, i ) ;
// Копіювання особи з області заданої координатами на вихідному зображенні
    //
    cvSetImageROI ( pFrame, cvRect ( r – > x, r – > y, r – > width, r
                  – > height )) ;
```

```
        imgFace.CopyOf ( pFrame ) ;
        cvResetImageROI ( pFrame ) ;
    // Обробка зображення обличчя imgFace
        //...
    }
```

Для виділення осіб використовується функція cvHaarDetectObjects другим параметром якої вказується навчений класифікатор.

Класифікатори поставляються разом з бібліотекою у вигляді xml файлів, в яких знаходиться інформація для сегментації зображення. Так само бібліотека дозволяє створювати свої класифікатори.

Так само в цьому блоці видаляється фон зображення. Але ця операція поки що не було реалізовано в проекті. Так як додатковий фон вносить перешкоди в систему, його видалення є необхідною умовою стабільної роботи системи.

Блок попередньої обробки

На вхід блоку попередньої обробки подається зображення особи з приділеною фоном. Під час попередньої обробки зображення масштабується до розміру 46x56 і перетворюється в складові сірого кольору.

Блок перетворення і виділення ключових ознак

У зв'язку з тим, що вхідні дані мають велику розмірність (46x56=2576 значень для одного зображення) було вирішено зменшити розмірність даних шляхом переходу і аналізу в інших просторах.

У проекті було обрано метод стиснення за допомогою перетворень Фур'є. Цей метод добре себе зарекомендував в області стиснення відео і зображень. Зокрема він використовується при стисненні даних у форматі JPEG.

Вся принадність у тому, що після стиснення зображення і подальшої реконструкції (зворотний процес стисненню) вихідне і отримане зображення схожі один з одним.

Це дозволяє не тільки значно прискорити швидкість навчання системи, а також внаслідок видалення дрібних перешкод (наприклад, дрібної зернистості), підвищити точність.

Однак при значному збільшенні ступеня стиснення, реконструйовані образи наближатися один до одного, що може викликати помилки другого роду. На підставі експериментів були обрані наступну кількість коефіцієнтів Фур'є:

– двошаровий персептрон – 200 коефіцієнтів ;

– NEFClass – 20 коефіцієнтів.

Розглянемо процес більш докладно. У процесі перетворення двомірне зображення перетворюється в одновимірний вектор по рядках.

Над вектором пікселів (значення яких становить [0;255]) здійснюється дискретне перетворення Фур'є для переходу в частотну область.

Після дискретного перетворення відбирається певна кількість перших значущих коефіцієнтів (200) і здійснюється зворотне перетворення.

Таким чином, зображення стискується з 2576 пікселів в 200 чисел. Для реконструкції цього зображення необхідно здійснити ДПФ, заповнити відсутні коефіцієнти ([202 ; 2576]) нулями і над 2576 коефіцієнтів Фур'є проробити операцію зворотного перетворення.

Так як після зворотного перетворення залишається комплексна складова для подальшої роботи береться її абсолютне значення.

Після виконаних операцій відбувається операція нормалізації вихідних даних – приведення значень в кордону 16-ричного цілого числа, тобто в область значень.

Алгоритм перетворення картинки до монохромного зображення

```
for Y := 0 to Height-1 do
for X := 0 to Width-1 do begin
  Color := Pixel[X,Y];
  Color := (Red(Color)+Green(Color)+Blue(Color)) div 3;
  if Color > 100 then Pixel[X,Y] := White
                 else Pixel[X,Y] := Black;
end;
```

Далі позбуваємося від смуг. Довідаємося їх цвіт, для чорних смуг сума чорних пікселів у кожному рядку або стовпцю завжди більше нуля.

```
Colorline := Black;
for X := 0 to Width-1 do begin
```

```
    Count := 0;
    for Y := 0 to Height-1 do if Pixel[X,Y] then Inc(Count);
    if Count = 0 then begin
      Colorline := White;
      Break;
    end;
  end;
```

Дані про вертикальні й горизонтальні лінії заносимо до масиву Vertical і Horizontal. Потім відновлюємо втрачений колір. Чорний піксель на лінії міняємо на білий, якщо у парі суміжних точок по вертикалі або горизонталі мають білий колір. Аналогічно діємо для білого пікселя.

Слід звернути увагу, що повністю відтворити зображення під лініями неможливо, символи можуть бути обрізані з кожної зі сторін або навпаки бути ширше й вище. Тому для успішної ідентифікації необхідно спочатку порівнювати, змістивши еталонне зображення на 1 піксель вліво, потім вправо, нагору й униз.

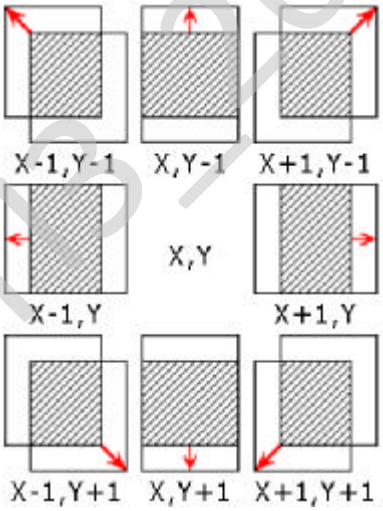

Рисунок 4.6 – Зсув картинки при порівнянні

В одному з 9 випадків результат порівняння буде найвищим 90% – 98%. Можна їх розділити посередині білою лінією, але надійніше виконати порівняння з еталоном по лівому краю.

Виявивши самий схожий символ, можна довідатися праву границю першого символу, а відповідно – ліву границю другого символу.

При такому підході можна розпізнати будь-яка кількість склеєних символів, з різною шириною.

Далі була створена канва для малювання й формування її образа в пам'яті. У якості канви використовувався клас Tbitmap (для простоти роботи з бітмапом використовуємо режим 1 байт на піксель, тобто Tbitmap.Pixelformat:=pf8bit), реалізуємо його на Tpaintbox, відображаємо в пам'яті за допомогою структури:

```
type
  Masx = Pbytearray;
var
Masy : array of Masx // масив пікселів,
{ де Masy[ y-координата][ x- координата] = номер кольору в палітрі
 цвіту (  при 8 біт/піксель). Відображення здійснюємо з
 використанням Tbitmap.Scanline (швидко й просто) }
  Setlength(Masy, Tbitmap.Height);
  for j := 0 to Tbitmap.Height - 1 do Masy[j] := Tbitmap.Scanline[j];
```

Для пояснення подальшого алгоритму дії необхідно розглянути просту математичну теорію.

Зрозуміло, для простоти розглядається вже тільки чорно-білі зображення. Нехай у нас малюнок полягає всього із двох пікселів. Тоді безліч усіх об'єктів, яке можна буде зобразити (універсальна безліч), складається із чотирьох об'єктів: (0,0), (0,1), (1,0), (1,1), де 1 – чорний піксель, 0 – білий.

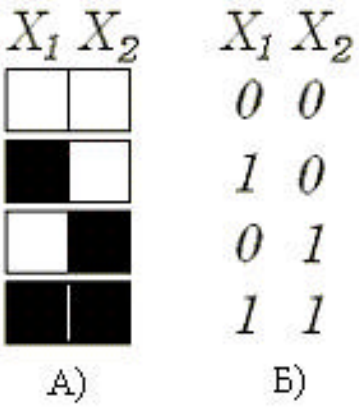

Рисунок 4.7 – Обробка зображення: а – чорно-біле зображення; б– переклад у двійкову представлення

Усі об'єкти універсальної множини можна розмістити у вершинах одиничного квадрата, таким чином, множини фігур, зображених на двох піксельному полі, може бути зіставлена множиною точок у двовимірному просторі. Ребру цього квадрата буде відповідати перехід від одного зображення до іншого.

Для переходу від (1,1) до (0,0) потрібно буде пройти два ребра, для переходу від (0,1) до (0,0) – одне. Відзначимо, що число ребер у нашому переході – це кількість незбіжних пікселів двох зображень.

Відстань від одного малюнка до іншого дорівнює числу незбіжних пікселів у них. Цю відстань називають відстанню по Хэммингу.

Як приклад представимо, що в нас малюнок складається із трьох пікселів. Коди зображень тоді будуть складатися із трьох значень, універсальна безліч – з восьми елементів, які ми розмістимо у вершинах одиничного куба. Але принципово нічого не зміниться, і відстань по Хеммингу обчислюється так само. У програмі використовується малюнок 50x70 = 3500 пікселів. Легко зміркувати, що в цьому випадку код будь-якого зображення складається з 3500 значень, універсальна безліч – з 23500 = 4,027 * 101053 елементів, які ми будемо розміщати у вершинах одиничного 3500 мірного куба. Уявити собі такий 3500 мірний куб нелегко, але зміст від цього не міняється абсолютно. Основна ідея полягає в тому, що в цьому багатомірному кубі зображення, що відповідають якомусь певному образу, лежать недалеко друг від друга.

Тепер можна показати як я це реалізувала в бакалаврській програмі. Після формування канви був створений і формований масив еталонних зразків символів. Еталонні зразки будемо формувалися на основі матриці розміром 16x16. Для цього була розроблена процедура генерації такої матриці по довільному зображенню еталона.

Процедура function Create_16x16(Img : Tbitmap): Tmas16x16 отримує в якості параметра посилання на картинку, на якій намальований еталон символу, повертає приведену матрицю розміром 16x16.

Коротко поясню роботу процедури в бакалаврській програмі.

1. Одержуємо посилання на бітмап і здійснюємо його відображення в пам'яті.

2. Обчислюємо координати границь (описаного прямокутника) образа (еталонного або розпізнаваного) шляхом сканування рядків/стовпців. При цьому тут і при подальшому аналізі зображення передбачається, що об’єкт намальований чорним кольором (№0 у палітрі квітів) і відповідно всі значущі пікселі мають значення 0.

```
for j := 0 to Img.Height - 1 do// Top
  begin
    for i := 0 to Img.Width - 1 do
      if Masy[j][i] = 0 then
      begin ytop := j; break; end;
    if ytop = j then break;
  end;
  for j := Img.Height - 1 downto 0 do
  begin
    for i := 0 to Img.Width - 1 do
      if Masy[j][i] = 0 then
      begin ybottom := j + 1; break; end;
    if ybottom = j + 1 then break;
  end;
  for i := 0 to Img.Width - 1 do
  begin
    for j := 0 to Img.Height - 1 do
      if Masy[j][i] = 0 then begin xleft := i; break; end;
    if xleft = i then break;
  end;
 for i := Img.Width - 1 downto 0 do
  begin
    for j := 0 to Img.Height - 1 do
if Masy[j][i] = 0 then begin xright := i + 1; break; end;
    if xright = i + 1 then break;
  end;
```

3. Для подальшого аналізу буде потрібно якийсь критерій, по якім буде проводитися згортка вихідного зображення символу в матрицю 16x16.

Таким критерієм був обраний загальний відсоток заповнення – відношення кількості значущих пікселів (з яких полягає символ) до загальної кількості пікселів в описаному навколо вихідного зображення прямокутнику. Даний параметр може впливати на якість розпізнавання, причому якщо він більше 1 для розпізнаваного символу буде відповідати менша кількість можливих альтернатив, при значенні меншому 1 – навпаки. У нашому випадку коефіцієнт виправлення прийняте рівним 0,99.

```
nsymbol := 0;
for j := ytop to ybottom do
for i := xleft to xright do
 if Masy[j][i] = 0 then inc(nsymbol);
Percent := nsymbol / ((ybottom - ytop)*(xright - xleft));
Percent := 0.99*Percent;
```

4. Далі розбиваємо прямокутник із зображенням символу на 16x16 комірок шляхом розподілу сторін нової комірки на 2. Запам'ятовуємо відносні координати кожної комірки й приступаємо до заповнення матриці 16x16. Приймаємо в якості критерію спільний відсоток заповнення. Якщо в аналізованій комірці відсоток заповнення більше, ніж загальний відсоток – відповідний елемент матриці 16x16 установлюється в 1, а якщо ні, то – в 0.

5. Інша частина алгоритму торкається питань малювання на Tbitmap букв або цифр (у циклі), запам'ятовування в масиві матриць 16x16, що відповідають кожному еталонному символу.

Після цього для розпізнавання здійснюємо порівняння матриці 16x16 шляхом розпізнаваного символу з матрицею еталона (шляхом перебору).

Порівняння робимо по елементне за допомогою оператора XOR. Результат – матриця 16x16, що містить одиниці в місцях розбіжностей тест – символу й еталона. Шляхом підрахунку кількості розбіжностей формуємо вектор, що містить цю інформацію для кожного еталонного символу, і робимо сортування його елементів по зростанню кількості розбіжностей.

Параметр (1 – Result[i]/256)*100%, де Result[i] – розбіжність для i – го символу, показує "імовірність" відповідності образа конкретному символу.

Для підвищення точності розпізнавання окремих символів, необхідно здійснювати додатковий аналіз признаків, наприклад симетричність образа (горизонтальна, вертикальна), наявність замкнених областей (ПРО, В, Д, Р и ін.), кількість відрізків і дуг, їх взаємне розташування й орієнтація (потрібно векторізація зображення).

Далі розглянемо завантаження шаблонів скриптов у бакалаврській програмі. При запуску програми в масив Data: array of array [0..9] of Tbitmap; записуються цифри від 0 до 9, написані наступними шрифтами: Arial, Century Gothic, Courier New Cyr, Goudy Old Style і Times New Roman – усього п'ять комплектів (можна легко збільшити).

```
Procedure Loaddata;
var i,j:integer; path:string;
begin
Setlength(Data,5);
for i := 0 to 4 do
begin
path := Extractfilepath(Application.Exename)+'\fonts\';
case i of
  0: path  := path + 'Arial\';
  1: path  := path + 'Century Gothic\';
  2: path  := path + 'Courier New Cyr\';
  3: path  := path + 'Goudy Old Style\';
  4: path  := path + 'Times New Roman\';
 end;
 for j := 0 to 9 do
  begin
  Data[i,j] := Tbitmap.Create;
Data[i,j].Loadfromfile(path + Inttostr(j) + '.bmp');
  end;
 end;
end;
```

Після завантаження еталонних зображень програма очікує одержання даних на обробку. При одержанні цих даних вираховуються відстані від розпізнаваного малюнка до кожного з еталонних (відстань по Хэммингу).

```
function Compare( b1,b2:Tbitmap):integer;
var i,j,count:integer;
```

```
begin
count := 0;
for i := 0 to 49 do
 for j := 0 to 69 do
if b1.Canvas.Pixels[i,j] <> b2.Canvas.Pixels[i,j] then
   inc(count);
   Result := count;
   end;
```

Знаючи цю відстань, легко обчислити потенціал, створюваний кожним еталонним малюнком у точці, що відповідає намальованому користувачем зображенню.

Потенціали, створювані нулями всіх накреслень, підсумовуються в p[0], одиницями – у p[1] і так далі.

```
for i := 0 to 4 do
  for j := 0 to 9 do
   begin
   r := Compare(Image1.Picture.Bitmap,Data[i,j]);
   p[j] := p[j] + 1000000/(1+r*r);
   end;
```

Після всього цього залишається знайти, якому образу відповідає найбільший потенціал.

Запропонований метод дає досить непогані результати як на машинописних, так і на рукописних цифрах. Алгоритм легко може бути переучений для розрізнення інших символів (латинських/руських/ українських букв, розділових знаків і т.п.).

Для підвищення надійності розпізнавання можна передбачити еталони, що сильно різняться між собою (для того, щоб якнайсильніше розкидати еталонні точки в межах компактної множини).

Для розпізнавання цифр я ще використав такий прийом: вихідний малюнок розбивався на 7 ділянок по вертикалі й 5 по горизонталі, ділянку цілком заливався білим або чорним залежно від того, яких пікселів ділянка містила більше, після чого робота з кожною ділянкою відбувалася як з пікселем.

4.2 Захист розробленого програмного забезпечення

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

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

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

3. Для шифрування 10-байтового блоку:

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

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

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

d. Повторюють етапи a-c для M2.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього видно, що інтерфейс користувача дуже простий та інтуїтивно зрозумілий. Після завантаження операційної системи, проходе завантаження розробленого ПЗ та включення фронтальної камери смартфону.

Далі проходе циклічне сканування відеопотоку з ціллю розпізнавання обличчя. При знаходженні обличчя проходе автоматичне порівняння та розпізнавання користувача. Якщо користувача розпізнано та йому дозволено використовувати смартфон проходе завершення роботи та передача управління операційній системі.

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено форму авторського права. Обрано shareware ліцензію (умовно-безплатне ПЗ). Авторське право є ключовою галуззю права інтелектуальної власності; воно призначене захищати лише зовнішню форму вираження об'єкта, тобто їхнє матеріальне втілення. Авторське право не може використовуватись для захисту абстрактних ідей, концепцій, фактів, стилів та технік, що можуть бути використані у творі. Захист авторського права – одна з важливих категорій теорії цивільного та цивільно-процесуального права. Під захистом авторських прав слід розуміти передбачені законом заходи із їхнього визнання, припинення їхнього порушення, застосування до правопорушників заходів юридичної відповідальності. Захист особистих немайнових і майнових прав суб'єктів авторського права здійснюється в порядку, встановленому адміністративним, цивільним і кримінальним законодавством. Було обрано Shareware умову розповсюдження.


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Центральноукраїнський національний технічний університет
Кафедра кібербезпеки та програмного забезпечення
ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему: Програмне забезпечення системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону"
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
Виконав: Павлюк В.Л.
Науковий керівник: Сичкова Т.В.
Кропивницький - 2024

Рисунок 5.2 – Розробник ПЗ

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем біометричної ідентифікації обличчя користувача смартфону.

– Досліджена система біометричної ідентифікації обличчя користувача смартфону.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання біометричної ідентифікації обличчя користувача смартфону.

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 призначені для системи кібербезпеки біометричної ідентифікації обличчя користувача

смартфону. 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 кібербезпеки й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Android.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм REDOC III.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.

2. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

3. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

4. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

5. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

6. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

7. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

8. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

9. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

10. Kuznetsov O., Frontoni E., Kuznetsova Ye., Smirnov O., Chevardin V. «Achieving Enhanced Security in Biometric Authentication: A Rigorous Analysis of Code-Based Fuzzy Extractor». *CEUR Workshop Proceedings,* Volume 3624, 2023, pp. 330-339.

11. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

12. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

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., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

15. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

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., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

19. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

20. 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.

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

22. 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.

23. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

24. 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.

25. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

26. 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.

27. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

28. 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.

29. 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.

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

31. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

32. 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.

33. 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.

34. 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.

35. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

36. 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.

37. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

38. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

39. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

40. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

41. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and*

Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.399-405.

42. 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.

43. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

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

45. 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.

46. 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.

47. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

48. Smirnov A.A., Kuznetsov 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.

49. Смірнов О.А. Козлов Я.О., Смірнова Т.В. «Дослідження застосування SIEM-систем для забезпечення кібербезпеки та захисту інформації». *II Міжнародна науково-практична Інтернет-конференція «Інновації та перспективні шляхи hозвитку інформаційних технологій (ІПШРІТ-2023)»* м.Черкаси 6 грудня 2023 року – Черкаси: ЧДТУ.– 2023. – С.251-252.

50. Козлов Я.О., Смірнова Т.В., Смірнов О.А. «Дослідження SIEM-систем для забезпечення кібербезпеки». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 26.

51. Козлов Я.О., Козірова Н.Л., Смірнов О.А. «Дослідження структури та принципу роботи SIEM-системи». *VII міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології" до 30-ти річчя кафедри кібербезпеки та програмного забезпечення,* м. Кропивницький. 1 листопада 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 59.

52. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 136-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) 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 кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону;

– 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 мобільному пристрої під керуванням ОС Android 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 Android.

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 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 71 аркуш.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 8.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнова Т.В.

Програмне забезпечення системи кібербезпеки біометричної ідентифікації обличчя користувача смартфону

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 35

Літера: РП

Кропивницький – 2024 року

ФАЙЛ ДАНИХ РОЗРОБНИКА - MYBOX.PAS

```
unit MyBox;
// Розробив студент Павлюхін Владислав Леонідович
// у ЦНТУ, кафедра КбПЗ, 2024 рік
// гр. КБ-21-3СК

interface // Секція інтерфейсу (зовнішніх визначень модуля)

Uses // Підключення бібліотек
  Windows, Messages, SysUtils,
  Variants, Classes, Graphics,
  Controls, Forms, Dialogs, StdCtrls, ExtCtrls;

Type // Визначає нову категорію
  TfrmAbout = class(TForm) // Визначення класу від TForm
    Image1: TImage; // Компонент зображення
    ButtonOk: TButton; // компонент кнопки
    lbComponentName: TLabel; // компонент текстових даних
    Label2: TLabel; // компонент текстових даних
    Label3: TLabel; // компонент текстових даних
    Label4: TLabel; // компонент текстових даних
    lbVersion: TLabel; // компонент текстових даних
    Label6: TLabel; // компонент текстових даних
    Label5: TLabel; // компонент текстових даних
    Label7: TLabel; // компонент текстових даних
  private
    { Private declarations }
  public
    { Public declarations }
    procedure ShowAbout(ComponentName: String; Version: String);
  end;

Var // Розділ визначення змінних
  frmAbout: TfrmAbout;

implementation // Секція реалізації

{$R *.dfm} // Підключення ресурсів

procedure TfrmAbout.ShowAbout(ComponentName: String; Version: String);
begin
  lbComponentName.Caption := ComponentName;
  lbVersion.Caption := Version;
  ShowModal;
end;

end.
```

ОСНОВНИЙ СБОРОЧНИЙ ФАЙЛ РОЗРОБЛЕНОЇ СИСТЕМИ - PROJECT.DPR

```
Program Project; // Початок файлу проекту ПЗ
// Розробив студент Павлюхін Владислав Леонідович
// у ЦНТУ, кафедра КбПЗ, 2024 рік
// гр. КБ-21-3СК
Uses // Підключення бібліотек
  Forms,  Windows,  Dialogs,  SysUtils,
  NNN1 in 'NNN1.pas' {frmMain},
  Mobile_Cam in ' Mobile_Cam.pas',
  CONVERT_VIDEO in 'CONVERT_VIDEO.pas',
  MyBox in 'MyBox.pas' {frmAboutBox };

{$R *.res} // Підключення ресурсів

Var // Розділ визначення змінних
  AppHandle : THandle;
  Buf    : String; // строкові дані

begin
  Buf    := ExtractFileName(UpperCase(ParamStr(0)));
  AppHandle := CreateMuteX(nil, False, PChar(Buf));
  if WaitForSingleObject(AppHandle, 0) <> WAIT_TIMEOUT then begin
    Application.Initialize; //Ініціалізіція ПЗ
// Заглавна назва
    Application.Title := 'Чекаю код…';
// Підключення форми frmMain
    Application.CreateForm(TfrmMain, frmMain);
// Підключення форми frmOption
    Application.CreateForm(TfrmOption, frmOption);
    Application.Run;
// Виведення головного вікна ПЗ
    end
  else begin
    ShowMessage('ПЗ завантажено'); // сповіщення
    end;
end. // Кінець файлу проекту ПЗ
```

ФАЙЛ ГОЛОВНОЇ ФОРМИ СИСТЕМИ - NNN1.PAS

```
unit NNN1;
// Розробив студент Павлюхін Владислав Леонідович
// у ЦНТУ, кафедра КбПЗ, 2024 рік
// гр. КБ-21-3СК
interface // Секція інтерфейсу (зовнішніх визначень модуля)

Uses // Підключення бібліотек
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ScrCam, ExtCtrls, ComCtrls, XPMan, Buttons;

Type // Визначає нову категорію
TfrmMain = class(TForm) // Визначення класу від TForm
    RecBut: TButton; // компонент кнопки
    PauseBut: TButton; // компонент кнопки
    AudVidOptBut: TButton; // компонент кнопки
    CheckBox1: TCheckBox; // компонент введення (так/ні)
    CheckBox3: TCheckBox; // компонент введення (так/ні)
    Edit1: TEdit; // компонент текстового введення даних
    Edit2: TEdit; // компонент текстового введення даних
    Edit3: TEdit; // компонент текстового введення даних
    Edit4: TEdit; // компонент текстового введення даних
    Label2: TLabel; // компонент текстових даних
    Label3: TLabel; // компонент текстових даних
    Label4: TLabel; // компонент текстових даних
    Label5: TLabel; // компонент текстових даних
    CheckBox4: TCheckBox; // компонент введення (так/ні)
    CheckBox5: TCheckBox; // компонент введення (так/ні)
    CheckBox6: TCheckBox; // компонент введення (так/ні)
    CheckBox8: TCheckBox; // компонент введення (так/ні)
    GroupBox1: TGroupBox;
    Edit5: TEdit; // компонент текстового введення даних
    UpDown1: TUpDown;
    UpDown2: TUpDown;
    Edit6: TEdit; // компонент текстового введення даних
    UpDown3: TUpDown;
    Edit7: TEdit; // компонент текстового введення даних
    Label6: TLabel; // компонент текстових даних
    Label7: TLabel; // компонент текстових даних
    Label8: TLabel; // компонент текстових даних
    RegSelect: TRadioGroup;
    ExitBut: TButton; // компонент кнопки
    Panel1: TPanel;
    Image1: TImage; // Компонент зображення
    Label9: TLabel; // компонент текстових даних
    ComboBox1: TComboBox;
    Label10: TLabel; // компонент текстових даних
    ComboBox2: TComboBox;
    Label11: TLabel; // компонент текстових даних
    XPManifest1: TXPManifest;
    Edit8: TEdit; // компонент текстового введення даних
    Label12: TLabel; // компонент текстових даних
    Panel2: TPanel;
    Progress: TProgressBar;
    BCancel: TButton; // компонент кнопки
    LStatus: TLabel; // компонент текстових даних
    UpDownTop: TUpDown;
    UpDownLeft: TUpDown;
    UpDownWidth: TUpDown;
    UpDownHeight: TUpDown;
    ComboBox3: TComboBox;
    Label13: TLabel; // компонент текстових даних
    GroupBox2: TGroupBox;
    VideoCodec: TLabel; // компонент текстових даних
    ElapsedTime: TLabel; // компонент текстових даних
    RealCapturing: TLabel; // компонент текстових даних
    CurrentCaptured: TLabel; // компонент текстових даних
    FramesCaptured: TLabel; // компонент текстових даних
```

```
DropedFrames: TLabel; // компонент текстових даних
CheckBox9: TCheckBox; // компонент введення (так/ні)
CheckBox2: TCheckBox; // компонент введення (так/ні)
BitBtn1: TBitBtn; // компонент - кнопка
SaveDialog: TSaveDialog;
FilterEffectsBut: TButton; // компонент кнопки
UpDown4: TUpDown;
Edit9: TEdit; // компонент текстового введення даних
Label1: TLabel; // компонент текстових даних
CheckBox7: TCheckBox; // компонент введення (так/ні)
Label14: TLabel; // компонент текстових даних
ComboBox4: TComboBox;
CheckBox10: TCheckBox; // компонент введення (так/ні)
VideoCodecValue: TLabel; // компонент текстових даних
ElapsedTimeValue: TLabel; // компонент текстових даних
RealCapturingValue: TLabel; // компонент текстових даних
CurrentCapturedValue: TLabel; // компонент текстових даних
FramesCapturedValue: TLabel; // компонент текстових даних
DropedFramesValue: TLabel; // компонент текстових даних
ScrCamera: TScreenCamera; // робота з камерою мобільного телефона
StopBut: TButton; // компонент кнопки
procedure DeleteSelectedfromList1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure StopButClick(Sender: TObject);
procedure RecButClick(Sender: TObject);
procedure AudVidOptButClick(Sender: TObject);
procedure CheckBox1Click(Sender: TObject);
procedure CheckBox3Click(Sender: TObject);
procedure CheckBox4Click(Sender: TObject);
procedure CheckBox5Click(Sender: TObject);
procedure CheckBox6Click(Sender: TObject);
procedure CheckBox8Click(Sender: TObject);
procedure CheckBox9Click(Sender: TObject);
procedure Edit5Change(Sender: TObject);
procedure RegSelectClick(Sender: TObject);
procedure ExitButClick(Sender: TObject);
procedure ComboBox1Change(Sender: TObject);
procedure ComboBox2Change(Sender: TObject);
procedure BCancelClick(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure Edit1KeyPress(Sender: TObject; var Key: Char);
procedure ComboBox3Change(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure Edit3Change(Sender: TObject);
procedure Edit4Change(Sender: TObject);
procedure CheckBox2Click(Sender: TObject);
procedure BitBtn1Click(Sender: TObject);
procedure FilterEffectsButClick(Sender: TObject);
procedure Edit9Change(Sender: TObject);
procedure Edit7KeyPress(Sender: TObject; var Key: Char);
procedure CheckBox7Click(Sender: TObject);
procedure ComboBox4Change(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure CheckBox10Click(Sender: TObject);
procedure ScrCameraUpdate(Sender: TObject);
procedure ScrCameraError(Sender: TObject; ErrorMessage: String);
procedure ScrCameraStart(Sender: TObject);
procedure ScrCameraStop(Sender: TObject);
procedure ScrCameraPreview(Sender: TObject; PreviewBitmap: TBitmap;
  Preview, Recording: Boolean);
procedure ScrCameraSaving(Sender: TObject; Percent: Integer;
  StatusCaption: String; var Continue: Boolean);
procedure ScrCameraDeleting(Sender: TObject; Percent: Integer;
  StatusCaption: String; var Continue: Boolean);
procedure ScrCameraOverlay(Sender: TObject; HDCBitmap: HDC; bmpWidth,
  bmpHeight: Integer);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure PauseButClick(Sender: TObject);
procedure ScrCameraPause(Sender: TObject);
```

```
    procedure ScrCameraResume(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
    TimerON : TTimerRecord;
    AllowExit,
    FCancel : Boolean;
  end;

Var // Розділ визначення змінних
  frmMain: TfrmMain;

implementation // Секція реалізації

Uses // Підключення бібліотек
OptDlg, FilterEffects;

{$R *.dfm} // Підключення ресурсів
Uses // Підключення бібліотек
Math;

Type // Визначає нову категорію
  Tmas16x16 = array [0..15] of array [0..15] of byte;
// матриця (шаблон) розпізнавання 16x16

Var // Розділ визначення змінних
  Massimple16 : array of Tmas16x16;  // масив шаблонів

//========== процедура генерації матриці

function Create_16x16(Img : Tbitmap) : Tmas16x16;
Type // Визначає нову категорію
  Masx = Pbytearray;
Var // Розділ визначення змінних
  Masy : array of Masx; // Бітмап у пам'яті як масив (Y x X)
  j, i : integer;
  xleft, xright, ytop, ybottom : integer; // аблолютна координата образа
  ki, kj : integer;
  nsymbol : integer; // кількість пікселей
  Percent : double; // відсоток заповнення
  // відносні координати аналізованих комірок
  XY : array [0..16] of record x, y : integer end;
W, H : integer; // ширина й висота образу розпізнавання

begin
  Setlength(Masy, Img.Height); // виділяємо пам'ять під бітмап
  for j := 0 to Img.Height - 1 do // одержуємо відображення бітмапа в масиві
   // Masy[y - координата][x - координата] = знач. пікселя (x,y)
    Masy[j] := Img.Scanline[j];
  //-------- одержання координат границь образа
  // Masy[y][x] = 0 відповідає чорному кольору пікселя
  xleft := -1;  // ініціалізація
  xright := -1;
  ytop := -1;
  ybottom := -1;

  for j := 0 to Img.Height - 1 do
  begin
    for i := 0 to Img.Width - 1 do
      if Masy[j][i] = 0 then
      begin ytop := j; break; end;
    if ytop = j then break;
  end;

  for j := Img.Height - 1 downto 0 do
  begin
    for i := 0 to Img.Width - 1 do
```

```
      if Masy[j][i] = 0 then
      begin ybottom := j + 1; break; end;
    if ybottom = j + 1 then break;
  end;

  for i := 0 to Img.Width - 1 do
  begin
    for j := 0 to Img.Height - 1 do
      if Masy[j][i] = 0 then begin xleft := i; break; end;
    if xleft = i then break;
  end;

  for i := Img.Width - 1 downto 0 do
  begin
    for j := 0 to Img.Height - 1 do
      if Masy[j][i] = 0 then begin xright := i + 1; break; end;
    if xright = i + 1 then break;
  end;


  if ((ybottom - ytop)*(xright - xleft)) = 0 then   // якщо немає даних
  begin
    exit;
  end;

  //-------------------------------------------------------
  // одержуємо відсоток заповнення
  // як відношення кількості значимих пікселей до загального
  // кількості пікселей у границях образа
  // % буде необхідний при аналізі кожної комірки
  // у образі
  nsymbol := 0;
  for j := ytop to ybottom do
  for i := xleft to xright do
    if Masy[j][i] = 0 then inc(nsymbol);
  Percent := nsymbol / ((ybottom - ytop)*(xright - xleft));
  Percent := 0.99*Percent;
// коефіцієнт який впливає на формування матриці 16x16
// > 1 - ураховується менше значимих пікселей
// < 1 - ураховується більше значимих пікселей
// розбиваємо прямокутник образа на 16 рівних частин
// шляхом розподілу сторін на 2
// і одержуємо відносні координати кожної комірки
  W := xright - xleft;;
  XY[0].x  := 0;
  XY[16].x := W;
  XY[8].x  :=   XY[16].x div 2;
  XY[4].x  :=   XY[8].x div 2;
  XY[2].x  :=   XY[4].x div 2;
  XY[1].x  :=   XY[2].x div 2;
  XY[3].x  :=  (XY[4].x + XY[2].x) div 2;
  XY[6].x  :=  (XY[8].x + XY[4].x) div 2;
  XY[5].x  :=  (XY[6].x + XY[4].x) div 2;
  XY[7].x  :=  (XY[8].x + XY[6].x) div 2;
  XY[12].x :=  (XY[16].x + XY[8].x) div 2;
  XY[10].x :=  (XY[12].x + XY[8].x) div 2;
  XY[14].x :=  (XY[16].x + XY[12].x) div 2;
  XY[9].x  :=  (XY[10].x + XY[8].x) div 2;
  XY[11].x :=  (XY[12].x + XY[10].x) div 2;
  XY[13].x :=  (XY[14].x + XY[12].x) div 2;
  XY[15].x :=  (XY[16].x + XY[14].x) div 2;
  H := ybottom - ytop;
  XY[0].y  := 0;
  XY[16].y := H;
  XY[8].y  :=   XY[16].y div 2;
  XY[4].y  :=   XY[8].y div 2;
  XY[2].y  :=   XY[4].y div 2;
  XY[1].y  :=   XY[2].y div 2;
  XY[3].y  :=  (XY[4].y + XY[2].y) div 2;
```

```
  XY[6].y  :=  (XY[8].y + XY[4].y) div 2;
  XY[5].y  :=  (XY[6].y + XY[4].y) div 2;
  XY[7].y  :=  (XY[8].y + XY[6].y) div 2;
  XY[12].y :=  (XY[16].y + XY[8].y) div 2;
  XY[10].y :=  (XY[12].y + XY[8].y) div 2;
  XY[14].y :=  (XY[16].y + XY[12].y) div 2;
  XY[9].y  :=  (XY[10].y + XY[8].y) div 2;
  XY[11].y :=  (XY[12].y + XY[10].y) div 2;
  XY[13].y :=  (XY[14].y + XY[12].y) div 2;
  XY[15].y :=  (XY[16].y + XY[14].y) div 2;
  // аналізуємо кожну комірку у прямокутнику образа
  // і створюємо матрицю 16x16
  for kj := 0 to 15 do
  for ki := 0 to 15 do
  begin
    nsymbol := 0;
    for j := ytop + XY[kj].y to ytop + XY[kj+1].y do
// пробігаємося по комірках уже
    for i := xleft + XY[ki].x to xleft + XY[ki+1].x do
// в абсолютних координатах
      if Masy[j][i] = 0 then inc(nsymbol);
// значення 0 = чорний колір
    if nsymbol / MAX(1, ((XY[ki+1].x - XY[ki].x) * (XY[kj+1].y - XY[kj].y))) >
Percent
      then Result[kj][ki] := 1 else Result[kj][ki] := 0;
// результат - матриця 16x16
  end;
  Setlength(Masy, 0);
end;

procedure TMain1.Formcreate(Sender: Tobject);
begin
  Img := Tbitmap.Create;              // Створюємо бітмап
  Img.Pixelformat := pf8bit;
  Img.Width := 200;
  Img.Height := 200;
  Panpaint.DoubleBufed := true;
// Щоб не було мерехтіння на Paintbox
  Lfont.Caption := 'Name: ' + FD.Font.Name + #13#10 +
// інформація про шрифт шаблону
                   'Size: ' + inttostr(FD.Font.Size);
end;

procedure TMain1.Bfontclick(Sender: Tobject);
begin
  if FD.Execute then Lfont.Caption := 'Name: ' + FD.Font.Name + #13#10 +
                   'Size: ' + inttostr(FD.Font.Size);
end;

Var // Розділ визначення змінних
  isymbol : integer;      // допоміжна змінна для визначення
                          // типу шаблонів  1 - букви, 2 - цифри

procedure TMain1.Bcharsclick(Sender: Tobject);
Var // Розділ визначення змінних
  i : integer;
begin
  isymbol := 1;
  Setlength(Massimple16, 32*Sizeof(Tmas16x16));
// виділяємо пам'ять під букви від А до Я
  for i := 0 to 31 do
  with Img.Canvas do
  begin
    Brush.Color := clwhite; // очищаємо
    Pen.Color := clwhite;
    Rectangle(0,0,Img.Width,Img.Height);
    Pen.Color := clblack;
    Font.Color := clblack;
    Font.Size := FD.Font.Size;
```

```
    Font.Style := FD.Font.Style;
    Font.Name := FD.Font.Name;
    Img.Canvas.Textout(10, 10, CHR(ORD('A')+i));
    Massimple16[i] := Create_16x16(Img); // створюємо шаблон для i-того символу
  end;
  Bclear.Click; // очищаємо
  Messagebox(handle, 'Матриця шаблонів створена. Намалюйте образ для
            розпізнавання й натисніть "Аналіз".', 'OK!', MB_OK or
            MB_Iconinformation);
end;

procedure TMain1.Bnumericclick(Sender: Tobject);
Var // Розділ визначення змінних
  i : integer;
begin
  isymbol := 2;
  Setlength(Massimple16, 10*Sizeof(Tmas16x16));
// виділяємо пам'ять під цифри від 0 до 9
  for i := 0 to 9 do
  with Img.Canvas do
  begin
    Brush.Color := clwhite;
    Pen.Color := clwhite;
    Rectangle(0,0,Img.Width,Img.Height);
    Pen.Color := clblack;
    Font.Color := clblack;
    Font.Size := FD.Font.Size;
    Font.Style := FD.Font.Style;
    Font.Name := FD.Font.Name;
    Img.Canvas.Textout(10, 10, CHR(ORD('0')+i));
    Massimple16[i] := Create_16x16(Img);
  end;
  Bclear.Click;
  Messagebox(handle, 'Матриця шаблонів створена. Намалюйте образ для
            розпізнавання й натисніть "Аналіз".', 'OK!', MB_OK or
            MB_Iconinformation);
end;

procedure TMain1.Banalyzeclick(Sender: Tobject);
Var // Розділ визначення змінних
  k,i,j, ki, kj : integer;
  Mas,                          // 16x16 підсумкова матриця
  Maschar : Tmas16x16;          // 16x16 матриця мальованого образа
  Res : array [0..31] of byte;  // масив для кожного символу
  nmax : integer;
  imin : integer;
begin
  if Length(Massimple16) = 0 then
  begin
    Messagebox(handle, 'Спочатку потрібно створити матрицю шаблонів!',
'Помилка!', MB_OK or MB_ICONWARNING);
    exit;
  end;
  Maschar := Create_16x16(Img);    // одержуємо 16x16 матрицю образа
  PB16x16.Repaint;
  with PB16x16.Canvas do
  for kj := 0 to 15 do
  for ki := 0 to 15 do
  begin
    Brush.Color := clred;
    if Maschar[kj][ki] = 1 then Brush.Style := bssolid else Brush.Style :=
bsclear;
    Rectangle(ki * 7, kj * 7, ki * 7 + 7, kj * 7 + 7);
  end;
  for k := 0 to 31 do
// одержуємо 16x16 підсумкову матрицю для кожного символу
  begin
    for j := 0 to 15 do
    for i := 0 to 15 do
```

```
      Mas[j][i] := Maschar[j][i] xor Massimple16[k][j][i];
    Res[k] := 0;
    for j := 0 to 15 do
    for i := 0 to 15 do
      Res[k] := Res[k] + Mas[j][i];
  end;
  Lbresult.Clear;
  if isymbol = 1 then
// робимо сортування для букв
    for i := 0 to 31 do
    begin
      nmax := 255;
      imin := 0;
      for k := 0 to 31 do
      if Res[k] < nmax then
      begin
        imin := k;
        nmax := Res[k];
      end;
      // результат виводимо у вигляді відсотків
Lbresult.Items.Add(CHR(ORD('A') + imin) + '     ' + inttostr(round(100*(1 -
Res[imin] / 256)))+'%');
      Res[imin] := 255;
    end
  else
  if isymbol = 2 then
    for i := 0 to 9 do
    begin
      nmax := 255;
      imin := 0;
      for k := 0 to 9 do
      if Res[k] < nmax then
      begin
        imin := k;
        nmax := Res[k];
      end;
Lbresult.Items.Add(CHR(ORD('0') + imin) + '     ' + inttostr(round(100*(1 -
Res[imin] / 256)))+'%');
      Res[imin] := 255;
    end;
end;
end.

procedure TMain1.FormCreate(Sender: TObject);
begin
  AllowExit := True;
  FCancel   := False;
end;

procedure TMain1.StopButClick(Sender: TObject);
begin
  if ScrCamera.IsRecording then
    ScrCamera.StopRecording;
end;

procedure TMain1.RecButClick(Sender: TObject);
begin
  if Edit8.Text <> '' then
    ScrCamera.StartRecording(Edit8.Text);
end;

procedure TMain1.AudVidOptButClick(Sender: TObject);
begin
  frmOption.ShowModal;
end;

procedure TMain1.CheckBox1Click(Sender: TObject);
begin
  ScrCamera.RecordCursor := CheckBox1.Checked;
```

```
end;

procedure TMain1.CheckBox3Click(Sender: TObject);
begin
  ScrCamera.DrawAreaCapture := CheckBox3.Checked;
end;

procedure TMain1.CheckBox4Click(Sender: TObject);
begin
  ScrCamera.LineRectClear := CheckBox4.Checked;
end;

procedure TMain1.CheckBox5Click(Sender: TObject);
begin
  ScrCamera.MinimizeAppOnStart := CheckBox5.Checked;
end;

procedure TMain1.CheckBox6Click(Sender: TObject);
begin
  ScrCamera.RestoreAppOnStop := CheckBox6.Checked;
end;

procedure TMain1.CheckBox8Click(Sender: TObject);
begin
  RegSelectClick(Self);
  ScrCamera.ShowPreview := CheckBox8.Checked;
end;
procedure TMain1.PauseButClick(Sender: TObject);
begin
  if ScrCamera.IsRecording then
    if not ScrCamera.IsPaused then
      ScrCamera.PauseRecording
    else
      ScrCamera.ResumeRecording;
end;

procedure TMain1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
  if AllowExit then
    CanClose := True
  else
    CanClose := False;
end;


procedure TMain1.DeleteSelectedfromList1Click(Sender: TObject);
begin
 ListBox1.DeleteSelected;
end;

procedure TMain1.Button1Click(Sender: TObject);
Var // Розділ визначення змінних
 i: integer;
begin
 if not OpenPictureDialog1.Execute then
  exit;
 with OpenPictureDialog1 do
  for i := 0 to Files.Count - 1 do
   ListBox1.Items.add(Files.Strings[i]);
end;

procedure TMain1.Button2Click(Sender: TObject);
begin
 if not OpenDialog1.Execute then
  exit;
 Label9.Caption := OpenDialog1.filename;
 MediaPlayer1.Close;
 MediaPlayer1.filename := Label9.Caption;
 MediaPlayer1.open;
```

```
 MediaPlayer1.Timeformat := tfMilliseconds;
 Label2.Caption := IntToStr(round(1 / 1000 * MediaPlayer1.Length)) + ' sec';
end;

procedure TMain1.ComboBox1Change(Sender: TObject);
begin
 case ComboBox1.ItemIndex of
  0: AviWriter.PixelFormat := pf1Bit;
  1: AviWriter.PixelFormat := pf4Bit;
  2: AviWriter.PixelFormat := pf8bit;
  3: AviWriter.PixelFormat := pf24bit;
  4: AviWriter.PixelFormat := pf32Bit;
 end;
 UpdateCompressorList;
end;

procedure TMain1.UpdateCompressorList;
var
 Save: integer;
begin
 Save := ComboBox2.ItemIndex;
 AviWriter.Compressorlist(ComboBox2.Items);
 if (Save >= 0) and (Save < ComboBox2.Items.Count) then
  ComboBox2.ItemIndex := Save
 else
  ComboBox2.ItemIndex := 0;
end;

procedure TMain1.Button3Click(Sender: TObject);
begin
 if ComboBox2.ItemIndex > 0 then
  AviWriter.ShowCompressorDialog(Self);
end;

procedure TMain1.ComboBox2Change(Sender: TObject);
begin
 if ComboBox2.ItemIndex > 0 then
  AviWriter.SetCompression(copy(ComboBox2.Items.Strings[ComboBox2.ItemIndex], 1,
4))
 else
  AviWriter.SetCompression('');
end;

procedure TMain1.FormCreate(Sender: TObject);
begin
 UpdateCompressorList;
end;

procedure TMain1.Button4Click(Sender: TObject);
begin
 ListBox1.Clear;
end;

procedure TMain1.Button5Click(Sender: TObject);
begin
 if not SaveDialog1.Execute then
  exit;
 AviWriter.filename := SaveDialog1.filename;
 AviWriter.TempFileName :=
  ExtractFilePath(AviWriter.filename) + '~AWTemp' +
ExtractFileName(AviWriter.filename);
 Label10.Caption := ExtractFileName(AviWriter.filename);
end;

procedure TMain1.AviWriterProgress(Sender: TObject; FrameCount: integer;
 var abort: boolean);
begin
 ProgressBar1.position := FrameCount;
end;
```

```
procedure TMain1.Button6Click(Sender: TObject);
Var // Розділ визначення змінних
 i: integer;
 Pic: TPicture;
 bm: TBitmap;
begin
 if AviWriter.filename = '' then
 begin
  ShowMessage('Введіть ім'я файлу');
  exit;
 end;
 Animate1.Active := false;
 Animate1.filename := '';
 MediaPlayer1.Close;
 with AviWriter do
 begin
  Width := SpinEdit1.value;
  Height := SpinEdit2.value;
  FrameTime := SpinEdit3.value;
  if Label9.Caption = 'None' then
   WavFileName := ''
  else
   WavFileName := Label9.Caption;
  Stretch := CheckBox1.checked;
  OnTheFlyCompression := Checkbox2.checked;
 end;

 ProgressBar1.max := ListBox1.Items.Count;
 ProgressBar1.position := 0;
 Label11.Caption := '... Initializing cam';
 Label11.Refresh;
 Label11.Caption := '... Adding cam';
 Label11.Refresh;
 for i := 0 to ListBox1.Items.Count - 1 do
 begin
  Pic := TPicture.Create;
  try
   Pic.loadfromfile(ListBox1.Items.Strings[i]);
   bm := TBitmap.Create;
   try
    bm.PixelFormat := AviWriter.PixelFormat;
    bm.Width := Pic.Width;
    bm.Height := Pic.Height;
    bm.Canvas.draw(0, 0, Pic.Graphic);
    if i = 0 then
     AviWriter.InitVideo(bm);
    AviWriter.AddFrame(bm);
   finally
    bm.Free;
   end;
  finally
   Pic.Free;
  end;
 end;

 AviWriter.FinalizeVideo;

 Label11.Caption := '... Writing File';
 Label11.Refresh;
 AviWriter.WriteAvi;

 Label11.Caption := '... Done';
 Button7.enabled := true;
end;

procedure TMain1.Button8Click(Sender: TObject);
begin
 Label9.Caption := 'None';
```

```
 MediaPlayer1.Close;
end;

procedure TMain1.Button7Click(Sender: TObject);
begin
 MediaPlayer1.Close;
 Animate1.Active := false;
 Animate1.filename := '';
 if FileExists(AviWriter.filename) then
 begin
  MediaPlayer1.filename := AviWriter.filename;
  MediaPlayer1.open;
  MediaPlayer1.Play;
 end
 else
  ShowMessage('Avi-File не преобразовано');
end;

procedure TMain1.Button9Click(Sender: TObject);
begin
 MediaPlayer1.Close;
 Animate1.Active := false;
 Animate1.filename := '';
 if AviWriter.PixelFormat = pf24bit then
 if FileExists(AviWriter.filename) then
 begin
  Animate1.filename := AviWriter.filename;
  Animate1.Active := true;
 end
 else
  ShowMessage('Avi-File не преобразовано');
end;

procedure TMain1.ComboBox3Change(Sender: TObject);
begin
 case ComboBox3.ItemIndex of
  0: AviWriter.CompressionQuality := 10000;
  1: AviWriter.CompressionQuality := 9000;
  2: AviWriter.CompressionQuality := 8000;
  3: AviWriter.CompressionQuality := 6000;
  4: AviWriter.CompressionQuality := 4000;
 else
  AviWriter.CompressionQuality := 10000;
 end;
end;

end.
```

ФАЙЛ РОБОТИ КАМЕРОЮ МОБІЛЬНОГО ПРИСТРОЮ - MOBILE_CAM.PAS

```
unit Mobile_Cam;
// Розробив студент Павлюхін Владислав Леонідович
// у ЦНТУ, кафедра КбПЗ, 2024 рік
// гр. КБ-21-3СК
interface // Секція інтерфейсу (зовнішніх визначень модуля)

Uses // Підключення бібліотек
scVfw,
scFlashWnd,
scFreeHandWnd,
scSelObjWnd,
scConsts,
scHighTimer,
scEffects,
scWaveMixer,
scWaveUtils,
scWaveRecorders,
Windows, Messages,
SysUtils, Variants,
Classes, Graphics,
Controls, Forms,
Dialogs, StdCtrls,
ExtCtrls, MMSystem,
JPEG, Math;

Type // Визначає нову категорію
TSCEvent = procedure(Sender: TObject) of object;
TSCErrorEvent = procedure(Sender: TObject;
ErrorMessage: string) of object;
TPreviewEvent = procedure(Sender: TObject; PreviewBitmap: TBitmap;
Preview: Boolean; Recording: Boolean) of object;
TSaveEvent = procedure(Sender: TObject; Percent: Integer;
StatusCaption: String; var Continue: Boolean) of object;
TOverlayEvent = procedure(Sender: TObject; HDCBitmap: HDC;
bmpWidth, bmpHeight: Integer) of object;
TRecordcam = class;
TTimerRecord = record
TimerON : Boolean;
Hour    : Byte;
Min     : Byte;
Sec     : Byte;
end;
THMSM = record
Hour        : Integer;
Minute      : Integer;
Second      : Integer;
MilliSecond : Integer;
end;
TGetInfo = record
InputNames   : TStringList;
InputIndex   : Integer;
InputVolume  : Integer;
InputEnabled : Boolean;
end;
TScreenRegion = (SelObject, FreeHand, FixedMoving, FixedStable, FullScreen);
TScreenRotate = (R90D, R180D, R270D, Mirror0D, Mirror180D);
TOperation = (None, Success, Fail);
TICINFOS = array[0..50] of TICINFO;

TCamera = class(TComponent)
private
FCurrentCapturedFPS,
FRealCapturingFPS     : Extended;
FourCC,
CompfccHandler        : DWORD;
FOnResume             : TSCEvent;
FOnError              : TSCErrorEvent;
```

```
FOnPreview            : TPreviewEvent;
FOnSaving,
FOnDeleting           : TSaveEvent;
FOnOverlay            : TOverlayEvent;
FVideoCompressorInfo  : TICINFOS;
FFrame                : TFlashingWnd;
FPreviewTimer         : THighTimer;
FCursorPos            : TMouse;
FRecordAVIThread      : TRecordcam;
nColors               : TPixelFormat;
FVideoCompressorCount : Integer;
FPriority             : TThreadPriority;
FFreeHandMode         : Boolean;
FRegColor,
FRegColor1,
FRegColor2            : TColor;
FVideoCodecList,
FFormatList      : TStringList;
FTimerRecord          : TTimerRecord;
StrCodec,
FElapsedTime          : String;
FWndHandle            : HWnd;
FScreenRegion         : TScreenRegion;
FScreenRotate         : TScreenRotate;
FInfo            : TGetInfo;
TempCompressed        : APAVIStream;

function GetNColors: TPixelFormat;
function GetPriority: TThreadPriority;
function GetInterval: Integer;
function GetUpdateRate: Integer;
function GetQuality: Integer;
function GetRecord_MSPF: Integer;
function GetFilterCopy: Integer;
function GetScreenWidth: Integer;
function GetScreenHeight: Integer;
function GetLineRectClear: Boolean;
function GetShowPreview: Boolean;
function HBitmap2DDB(HBitmap: HBitmap; nBits: LongWord): THandle;
function MilliSecond2Time(TimeAtMillisecond: Extended): THMSM;
function GetVersion: String;
public
// конструктор класу
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
procedure SetInputIndex(Index: Integer);
procedure StopRecording;
procedure PauseRecording;
procedure ResumeRecording;
procedure CompressorConfigure(Compressor: Byte; WND: HWND);
procedure SetSizeFullScreen;
procedure GetMinimumScreenSize(var W, H: Integer);
procedure GetMaximumScreenSize(var W, H: Integer);
function StartRecording(szFileName: String): Boolean;
end;

TRecordcam = class(TThread)
private
      FScrCam: TCamera;
protected
      procedure Execute; override;
public
      constructor Create(ScrCam: TCamera);
end;

implementation // Секція реалізації

Var // Розділ визначення змінних
OldTime2           : Extended;
```

```
FSuccess           : TOperation;
SC                 : TCamera;

procedure FreeFrame(var alpbi: PBitmapInfoHeader);
begin
  if alpbi <> nil then begin
     GlobalFreePtr(alpbi);
    alpbi := nil;
end;
end;

procedure TCamera.ConvertToGrayScale(var AnImage: TBitmap);
Var // Розділ визначення змінних
BMPImage  : TBitmap;
JPGImage  : TJPEGImage;
MemStream : TMemoryStream;
begin
BMPImage := TBitmap.Create;
try
BMPImage.Width  := AnImage.Width;
BMPImage.Height := AnImage.Height;
JPGImage := TJPEGImage.Create;
try
JPGImage.Assign(AnImage);
JPGImage.CompressionQuality := 100;
JPGImage.Compress;
JPGImage.Grayscale := True;
BMPImage.Canvas.Draw(0, 0, JPGImage);
if Assigned(BMPImage) then
  BMPImage.PixelFormat := nColors;
MemStream := TMemoryStream.Create;
try
  BMPImage.SaveToStream(MemStream);
  MemStream.Position := 0;
  AnImage.LoadFromStream(MemStream);
finally
  MemStream.Clear;
  MemStream.Free;
end;
finally
JPGImage.Free;
end;
finally
BMPImage.Free;
end;
end;

function TCamera.GetVersion: String;
begin
Result := isVersion;
end;

procedure TCamera.SetVersion(Value: String);
begin
Value := Value;
end;

function TCamera.GetBrightness: Integer;
begin
Result := FBrightness;
end;

procedure TCamera.SetBrightness(Value: Integer);
begin
if FBrightness <> Value then
FBrightness := Value;
end;

function TCamera.GetContrast: Integer;
```

```
begin
Result := FContrast;
end;

procedure TCamera.SetContrast(Value: Integer);
begin
if FContrast <> Value then
FContrast := Value;
end;

function TCamera.GetNColors: TPixelFormat;
begin
Result := nColors;
end;

procedure TCamera.SetNColors(Value: TPixelFormat);
begin
nColors := Value;
case nColors of
pf8bit  : Bits := 8;
pf15bit,
pf16bit :
begin
  nColors := pf16bit;
  Bits    := 16;
end;
pf24bit : Bits := 24;
pf32bit : Bits := 32;
else
begin
  nColors := pf32bit;
  Bits    := 32;
end;
end;
end;

function TCamera.GetPriority: TThreadPriority;
begin
Result := FPriority;
end;

procedure TCamera.SetPriority(Value: TThreadPriority);
begin
if FPriority <> Value then begin
FPriority := Value;
if Assigned(FRecordAVIThread) then
FRecordAVIThread.Priority := FPriority;
end;
end;

function TCamera.GetInterval: Integer;
begin
Result := FPlaybackFPS;
end;

procedure TCamera.SetInterval(Value: Integer);
begin
if (FPlaybackFPS <> Value) and
(Value <= 1000) and
(Value > 0) then
FPlaybackFPS := Value;
end;

function TCamera.GetUpdateRate: Integer;
begin
Result := FUpdateRate;
end;

procedure TCamera.SetUpdateRate(Value: Integer);
```

```
begin
if (FUpdateRate <> Value) and
(Value <= 500) and
(Value > 0) then
FUpdateRate := Value;
end;

function TCamera.GetQuality: Integer;
begin
Result := FCompressionQuality;
end;

procedure TCamera.SetQuality(Value: Integer);
begin
if (FCompressionQuality <> Value) and
(Value <= 10000) and
(Value > 0) then
FCompressionQuality := Value;
end;

function TCamera.GetRecord_MSPF: Integer;
begin
Result := FMSPFRecord;
end;

procedure TCamera.SetRecord_MSPF(Value: Integer);
begin
if (FMSPFRecord <> Value) and
((Value <= 1000) and
(Value > 0)) then
FMSPFRecord := Value;
end;

function TCamera.GetFilterCopy: Integer;
begin
Result := FFilterCopy;
end;

procedure TCamera.SetFilterCopy(Value: Integer);
begin
if (FFilterCopy <> Value) and
((Value <= 1) and
(Value >= 0)) then
FFilterCopy := Value;
end;

function TCamera.GetScreenWidth: Integer;
begin
Result := FScreenWidth;
end;

procedure TCamera.SetScreenWidth(Value: Integer);
begin
if (FScreenWidth <> Value) then
FScreenWidth := Value;
end;

function TCamera.GetScreenHeight: Integer;
begin
Result := FScreenHeight;
end;

procedure TCamera.SetScreenHeight(Value: Integer);
begin
if (FScreenHeight <> Value) then
FScreenHeight := Value;
end;

function TCamera.GetLineRectClear: Boolean;
```

```
begin
Result := FLineRectClear;
end;

procedure TCamera.SetLineRectClear(Value: Boolean);
begin
if (FLineRectClear <> Value) then
FLineRectClear := Value;
end;

function TCamera.GetShowPreview: Boolean;
begin
Result := not FShowPreview;
end;

procedure TCamera.SetShowPreview(Value: Boolean);
begin
OldUpdateTime1 := 0;
OldTime2       := 0;
InitialTime1   := TimeGetTime;
if FShowPreview <> Value then
FShowPreview := Value;
if FShowPreview then begin
if not Assigned(FPreviewTimer) then begin
FPreviewTimer := THighTimer.Create(Self);
FPreviewTimer.OnTimer  := FShowPreviewTimer;
FPreviewTimer.Interval := 1000 div FPlaybackFPS;
FPreviewTimer.ThreadPriority := FPriority;
FPreviewTimer.Synchronize := True;
FPreviewTimer.UseThread := True;
end;
FPreviewTimer.Enabled := False;
if Assigned(FFrame) then
ShowWindow(FFrame.Handle, SW_HIDE);
FPreviewTimer.Enabled := True;
end
else begin
if not RecordState then begin
if Assigned(FPreviewTimer) then
  FPreviewTimer.Enabled := False;
if Assigned(FFrame) then
  ShowWindow(FFrame.Handle, SW_HIDE);
if Assigned(FOnPreview) then
  FOnPreview(Self, nil, False, False);
end;
end;
end;

procedure TCamera.CompressorHasFeatures(Compressor: Byte;
var hasAbout: Boolean; var hasConfig: Boolean);
Var // Розділ визначення змінних
icv: hic;
begin
hasAbout  := False;
hasConfig := False;
if Compressor >= FVideoCompressorCount then Exit;
  icv := ICOpen(FVideoCompressorInfo[Compressor].fccType,
          FVideoCompressorInfo[Compressor].fccHandler,
          ICMODE_QUERY);
  if (icv <> 0) then
begin
    hasAbout  := ICQueryAbout(icv);
    hasConfig := ICQueryConfigure(icv);
ICClose(icv);
end;
end;

function TCamera.GetVideoCompressorsInfo: TStringList;
Var // Розділ визначення змінних
```

```
ICV    : HIC;
   Source : PBitmapInfoHeader;
I      : Integer;
mImage : TBitmap;
begin
mImage := TBitmap.Create;
try
Source := CaptureScreenFrame(1,
                            mImage,
                            FScreenLeft,
                            FScreenTop,
                            FScreenWidth,
                            FScreenHeight,
                            FFilterCopy);
FVideoCompressorCount := 0;
FVideoCodecList.Clear;
for I := 0 to 50 do begin
       ICInfo(ICTYPE_VIDEO,
       I,
       @FVideoCompressorInfo[FVideoCompressorCount]);
      ICV := ICOpen(FVideoCompressorInfo[FVideoCompressorCount].fccType,
              FVideoCompressorInfo[FVideoCompressorCount].fccHandler,
              ICMODE_QUERY);
      if (ICV <> 0) then begin
            if (ICCompressQuery(ICV, Source, nil) = ICERR_OK) then begin
                ICGetInfo(ICV,
              @FVideoCompressorInfo[FVideoCompressorCount],
              SizeOf(TICINFO));
            Inc(FVideoCompressorCount);
    end;
            ICClose(ICV);
  end;
end;
for I := 0 to FVideoCompressorCount - 1 do
FVideoCodecList.Add(FVideoCompressorInfo[I].szDescription);
finally
mImage.Free;
end;
  FreeFrame(Source);
Result := FVideoCodecList;
end;

procedure TCamera.StopRecording;
begin
FSuccess := Success;
RecordState := False;
FPauseRecording := False;
if not FShowPreview then
SetShowPreview(False)
else
SetShowPreview(True);
end;

procedure TCamera.PauseRecording;
begin
FPauseRecording := True;
end;

procedure TCamera.ResumeRecording;
begin
if FPauseRecording then
FPauseRecording := False;
end;

procedure TCamera.ThreadDone(Sender: TObject);
begin
RecordState      := False;
FRecordAVIThread := nil;
end;
```

```
function SaveCallBack(nPercent: Integer): LONG; stdcall;
Var // Розділ визначення змінних
C : Boolean;
begin
Result := 0;
if Assigned(SC) then begin
if Assigned(SC.OnSaving) then begin
C := True;
SC.OnSaving(SC, nPercent, SavingMsg, C);
if not C then
  Result := -1;
end;
end;
end;

// конструктор класу TRecordcam
constructor TRecordcam.Create(ScrCam: TCamera);
begin
inherited Create(False);
FScrCam := ScrCam;
FreeOnTerminate := True;
end;

procedure TRecordcam.Execute;
Var // Розділ визначення змінних
Res      : Integer;
Variable : Boolean;
function IsDirectory(const DirName: string): Boolean;
var
Attr: Integer;
begin
Attr := SysUtils.FileGetAttr(DirName);
Result := (Attr <> -1) and (Attr and SysUtils.faDirectory =
SysUtils.faDirectory);
end;
begin
if IsDirectory(ExtractFilePath(FFileName)) then begin
repeat
Res := FScrCam.RecordVideo(FFileName);
until not (Res = -1);
Variable := True;
if SavingSuccess then
if Assigned(SC) then
  if Assigned(SC.OnSaving) then
    SC.OnSaving(SC, 100, SavingSuccessMsg, Variable);
end
else begin
if Assigned(SC) then
if Assigned(SC.OnError) then
  SC.OnError(SC, ErrorMsg49);
end;
PowerDeleteFile(TempVideoFile);
PowerDeleteFile(TempFile);
end;
end.
```

ФАЙЛ КОНВЕРТАЦІЇ ВІДЕО ДАНИХ НА ЛЬОТУ - CONVERT_VIDEO.PAS

```
unit CONVERT_VIDEO;
// Розробив студент Павлюхін Владислав Леонідович
// у ЦНТУ, кафедра КбПЗ, 2024 рік
// гр. КБ-21-3СК
interface // Секція інтерфейсу (зовнішніх визначень модуля)

Uses // Підключення бібліотек
Dialogs, Windows, Messages, SysUtils,
Classes, Graphics, Controls, Forms, BMP2AVI,
ActiveX;

////////////////////////////////
//  -cam for Windows        //
////////////////////////////////

Type // Визначає нову категорію

LONG = Longint;

APAVISTREAM = array[0..1] of PAVIStream;
APAVICompressOptions = array[0..1] of PAVICompressOptions;

TAVISaveCallback = function(i: integer): LONG; pascal;

Type // Визначає нову категорію
TFourCC = string[4];

Type // Визначає нову категорію
TProgressEvent = procedure(Sender: TObject; FrameCount: integer; var abort:
boolean) of object;

TBadBitmapEvent = procedure(Sender: TObject; bmp: TBitmap; InfoHeaderSize,
BitsSize: integer) of object;
Type // Визначає нову категорію
TAviWriter_2 = class(TComponent)
fFilename: string;
fWavFileName: string;
VideoStream: PAVIStream;
fPstream, fCompStream, fEditStream: PAVIStream;
fStreamInfo: TAviStreamInfo;
fFrameCount: integer;
fFourCC: TFourCC;
fPixelFormat: TPixelFormat;
fInHeader: TBitmapInfoHeader;
fPInInfo: PBitmapInfo;
fInInfoSize: integer;
AviCompressoptions: TAVICompressOptions;
fAbort: boolean;
fCompressionQuality: integer;
fInitialized, fFinalized: boolean;
fWaveFileList: TStringList;
fCompOnFly: boolean;

fOnProgress: TProgressEvent;
fOnBadBitmap: TBadBitmapEvent;

procedure AddVideo;
procedure SetWavFileName(value: string);
function AviSaveCallback(i: integer): LONG; pascal;
procedure SetPixelFormat(const value: TPixelFormat);
procedure InitStreamFormat(const bm: TBitmap);
procedure InternalAddFrame(const Bitmap: TBitmap; Key: boolean);
procedure SetHeight(const Value: integer);
procedure SetWidth(const Value: integer);
protected
public
Bitmaps: TList;
```

```
TempFileName: string;
SilenceName: string;
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
procedure Write;
procedure AddFrame(const ABmp: TBitmap);
procedure AddStillImage(const ABmp: TBitmap; Showtime: integer);
procedure FinalizeVideo;
procedure WriteAvi;
procedure Compressorlist(const List: TStrings);
procedure SetCompression(FourCC: TFourCC);
procedure SetCompressionQuality(q: integer);
procedure ShowCompressorDialog(ADialogParent: TWinControl);
procedure AddWaveFile(const filename: string; Delay: integer);
properties:
property Aborted: boolean read fAbort;
property OnTheFlyCompression: boolean read fCompOnFly write fCompOnFly;
property OnBadBitmap: TBadBitmapEvent read fOnBadBitmap write fOnBadBitmap;
property CompressionQuality: integer read FCompressionQuality write
FCompressionQuality;
published
property Height: integer read fHeight write SetHeight;
property Width: integer read fWidth write SetWidth;
property FrameTime: integer read fFrameTime write fFrameTime;
property Stretch: boolean read fStretch write fStretch;
property PixelFormat: TPixelFormat read fPixelFormat write SetPixelFormat;
property filename: string read fFilename write fFilename;
property WavFileName: string read fWavFileName write SetWavFileName;
property OnProgress: TProgressEvent read fOnProgress write fOnProgress;
end;

function AviSaveCallback(i: integer): LONG; pascal;
procedure Register;

const //введення констант
CP_PROGRESS = WM_USER + 50;

Var // Розділ визначення змінних
OwnerForm: TForm;

implementation // Секція реалізації

Uses // Підключення бібліотек
MMsystem, Silence;

Type // Визначає нову категорію
PICINFO = ^TICINFO;
TICINFO = packed record
dwSize: DWord;
fccType: DWord;
fccHandler: DWord;
dwFlags: DWord;
dwVersion: DWord;
dwVersionICM: DWord;
szName: array[0..15] of WChar;
szDescription: array[0..127] of WChar;
szDriver: array[0..127] of WChar;
end;
// конструктор класу Tcam_Avi
constructor Tcam_Avi.Create(AOwner: TComponent);
Var // Розділ визначення змінних
TempDir: string;
l: integer;
begin
inherited Create(AOwner);
Height := screen.Height div 10;
Width := screen.Width div 10;
fFrameTime := 1000;
fStretch := True;
```

```
fFilename := '';
Bitmaps := TList.Create;
AVIFileInit;

fFourCC := '';
fPixelFormat := pf24bit;
fAbort := false;
fCompressionQuality := 5000;
fCompOnFly := true;
fWaveFileList := TStringList.Create;
AVICreator := TAVICreator.Create;

SetLength(TempDir, MAX_PATH + 1);
l := GetTempPath(MAX_PATH, PChar(TempDir));
SetLength(TempDir, l);
if copy(TempDir, Length(TempDir), 1) <> '\' then
TempDir := TempDir + '\';
TempFileName := TempDir + '~AWTemp.avi';
end;

destructor Tcam_Avi.Destroy;
Var // Розділ визначення змінних
refcount: integer;
begin
AVICreator.Free;
Bitmaps.Free;
fWaveFileList.Free;
if fPInInfo <> nil then
FreeMem(fPInInfo);

if Assigned(pfile) then
try
repeat
refcount := AVIFileRelease(pfile);
until refcount <= 0;
except
pfile := nil;
end;

if Assigned(fCompStream) then
AVIStreamRelease(fCompStream);
if Assigned(fPstream) then
AVIStreamRelease(fPstream);
if Assigned(VideoStream) then
AVIStreamRelease(VideoStream);
if Assigned(Stream) then
AVIStreamRelease(Stream);

if FileExists(TempFileName) then
Deletefile(TempFileName);
AVIFileExit;
inherited;
end;

procedure Tcam_Avi.Write;
Var // Розділ визначення змінних
ExtBitmap: TBitmap;
nStreams: integer;
i: integer;
Streams: APAVISTREAM;
CompOptions: APAVICompressOptions;
AVIERR: HRESULT;
refcount: integer;
begin
Stream := nil;
VideoStream := nil;

for i := 0 to Bitmaps.Count - 1 do
begin
```

```
ExtBitmap := Bitmaps[i];
end;

try
AddVideo;
if WavFileName <> '' then
nStreams := 2
else
nStreams := 1;
Streams[0] := VideoStream;
Streams[1] := Stream;
CompOptions[0] := nil;
CompOptions[1] := nil;
AVIERR := AVISaveV(PChar(filename), nil, nil, nStreams, Streams, CompOptions);
finally
if Assigned(VideoStream) then
AVIStreamRelease(VideoStream);
if Assigned(Stream) then
AVIStreamRelease(Stream);
try
repeat
 refcount := AVIFileRelease(pfile);
until refcount <= 0;
except
end;

pfile := nil;
VideoStream := nil;
Stream := nil;

Deletefile(TempFileName);
end;
end;

procedure Tcam_Avi.AddVideo;
Var // Розділ визначення змінних
pstream: PAVIStream;
StreamInfo: TAviStreamInfo;
BitmapInfo: PBitmapInfoHeader;
BitmapInfoSize: integer;
BitmapSize: Longint;
BitmapBits: Pointer;
Bitmap: TBitmap;
ExtBitmap: TBitmap;
Samples_Written: LONG;
Bytes_Written: LONG;
AVIERR: integer;
i: integer;
begin
Bitmap := TBitmap.Create;
Bitmap.Height := Self.Height;
Bitmap.Width := Self.Width;

try
FillChar(StreamInfo, SizeOf(StreamInfo), 0);

StreamInfo.dwRate := 1000;
StreamInfo.dwScale := fFrameTime;
StreamInfo.fccType := streamtypeVIDEO;
StreamInfo.fccHandler := 0;
StreamInfo.dwFlags := 0;
StreamInfo.dwSuggestedBufSize := 0;
StreamInfo.rcFrame.Right := Self.Width;
StreamInfo.rcFrame.Bottom := Self.Height;
try
for i := 0 to Bitmaps.Count - 1 do
begin
 BitmapInfo := nil;
 BitmapBits := nil;
```

```
 try
  ExtBitmap := Bitmaps[i];
  if fStretch then
   Bitmap.Canvas.StretchDraw(Rect(0, 0, Self.Width, Self.Height), ExtBitmap)
  else
  try
   with Bitmap.Canvas do
   begin
    Brush.Color := ExtBitmap.Canvas.Pixels[0, 0];
    Brush.Style := bsSolid;
    FillRect(Rect(0, 0, Bitmap.Width, Bitmap.Height));
    draw(0, 0, ExtBitmap);
   end;
  except
   Bitmap.Canvas.StretchDraw(Rect(0, 0, Self.Width, Self.Height), ExtBitmap);
  end;
  InternalGetDIBSizes(Bitmap.Handle, BitmapInfoSize, BitmapSize, pf8bit);
  GetMem(BitmapInfo, BitmapInfoSize);
  GetMem(BitmapBits, BitmapSize);
  InternalGetDIB(Bitmap.Handle, 0, BitmapInfo^, BitmapBits^, pf8bit);

  AVIERR := AVIStreamWrite(pstream, i, 1, BitmapBits, BitmapSize,
AVIIF_KEYFRAME, Samples_Written, Bytes_Written);
  if AVIERR <> AVIERR_OK then
   raise Exception.Create
finally
  if (BitmapInfo <> nil) then
   FreeMem(BitmapInfo);
  if (BitmapBits <> nil) then
   FreeMem(BitmapBits);
 end;
end;
finally
AVIStreamRelease(pstream);
end;
finally
Bitmap.Free;
end;
end;
try
finally
AVIFileRelease(InputFile);
end;
end;

procedure Tcam_Avi.InitializeBitmapInfoHeader(Bitmap: HBITMAP; var Info:
TBitmapInfoHeader;
PixelFormat: TPixelFormat);

Var // Розділ визначення змінних
DIB: TDIBSection;
Bytes: integer;
function AlignBit(Bits, BitsPerPixel, Alignment: Cardinal): Cardinal;
begin
Dec(Alignment);
Result := ((Bits * BitsPerPixel) + Alignment) and not Alignment;
Result := Result shr 3;
end;
begin
DIB.dsbmih.biSize := 0;
Bytes := GetObject(Bitmap, SizeOf(DIB), @DIB);
else
begin
FillChar(Info, SizeOf(Info), 0);
with Info, DIB.dsBm do
begin
biSize := SizeOf(Info);
biWidth := bmWidth;
biHeight := bmHeight;
```

```
end;
end;
case PixelFormat of
pf1Bit: Info.biBitCount := 1;
pf4Bit: Info.biBitCount := 4;
pf8bit: Info.biBitCount := 8;
pf24bit: Info.biBitCount := 24;
end;
Info.biPlanes := 1;
Info.biCompression := BI_RGB;
Info.biSizeImage := AlignBit(Info.biWidth, Info.biBitCount, 32) *
Cardinal(abs(Info.biHeight));

end;

procedure Tcam_Avi.SetWavFileName(value: string);
begin
if LowerCase(fWavFileName) <> LowerCase(value) then
if value <> '' then
else
 fWavFileName := value
else
fWavFileName := value;
end;

procedure Tcam_Avi.InternalAddFrame(const Bitmap: TBitmap; Key: boolean);
Var // Розділ визначення змінних
Samples_Written: LONG;
Bytes_Written: LONG;
AVIERR: integer;
DIB: TDIBSection;
DIBErr: integer;
flag: DWord;
begin
if fFrameCount = 0 then
begin
InitStreamFormat(Bitmap);
end;

FillChar(DIB, SizeOf(DIB), 0);
DIBErr := GetObject(Bitmap.Handle, SizeOf(DIB), @DIB);
if DIBErr = 0 then
begin
end;

if Key then
flag := AVIIF_KEYFRAME
else
flag:=0;
try
AVIERR := AVIStreamWrite(fCompStream, fFrameCount, 1, DIB.dsBm.bmBits,
                         DIB.dsbmih.biSizeImage, flag,
                         Samples_Written, Bytes_Written);
except
   AVIERR := AVIERR_ERROR;
end;

inc(fFrameCount);

if Assigned(fOnProgress) then
if (fFrameCount mod 20 = 0) then
fOnProgress(Self, fFrameCount, fAbort);
end;

procedure Tcam_Avi.FinalizeVideo;
begin
fInitialized := false;
fFinalized := true;
end;
```

```
procedure Tcam_Avi.InitVideo(const ABmp: TBitmap);
Var // Розділ визначення змінних
S, Workfile: string;
AVIERR: HRESULT;
begin
if not fCompOnFly or (fFourCC = '') then
begin
if fFourCC = '' then
Workfile := fFilename
else
Workfile := TempFileName;
AVICreator.CreateMJPEGFile(Workfile, 0, 0, fFrameTime, 10000, ABmp, nil, 0);
if fFourCC = '' then
Exit;
end;
VideoStream := nil;
fCompStream := nil;
fPstream := nil;
Stream := nil;
fAbort := false;
pfile := nil;

if fCompOnFly then
begin
if fCompOnFly then
Workfile := fFilename;
else
Workfile := TempFileName;
if FileExists(Workfile) then

AVIERR := AVIFileOpen(pfile, PChar(Workfile), OF_WRITE or OF_CREATE, nil);

FillChar(fStreamInfo, SizeOf(fStreamInfo), 0);

fStreamInfo.dwRate := 1000;
fStreamInfo.dwScale := fFrameTime;
fStreamInfo.fccType := streamtypeVIDEO;
S := fFourCC;
if S = '' then
fStreamInfo.fccHandler := 0
else
fStreamInfo.fccHandler := mmioStringToFOURCC(PChar(S), 0);
fStreamInfo.dwQuality := fCompressionQuality;
fStreamInfo.dwFlags := 0;
fStreamInfo.dwSuggestedBufSize := 0;
fStreamInfo.rcFrame.Right := Self.Width;
fStreamInfo.rcFrame.Bottom := Self.Height;
end
else
begin
Workfile := TempFileName;
InitStreamFormat(ABmp);
end;
fFrameCount := 0;
fInitialized := true;
end;

procedure Tcam_Avi.WriteAvi;
Var // Розділ визначення змінних
nStreams: integer;
Streams: APAVISTREAM;
CompOptions: APAVICompressOptions;
AVIERR: HRESULT;
Refcount: integer;
begin
if not fCompOnFly or (fFourCC = '') then
begin
AVICreator.CloseMJPEGFile(false);
```

```
if fFourCC = '' then
Exit;
end;
if fAbort or (not fFinalized) then
begin
if fPstream <> nil then
AVIStreamRelease(fPstream);
if fCompStream <> nil then
AVIStreamRelease(fCompStream);
fCompStream := nil;
fPstream := nil;
fWaveFileList.Clear;
try
repeat
 refcount := AVIFileRelease(pfile);
until refcount <= 0;
pfile := nil;
except
pfile := nil;
end;
exit;
end;

try
if not fCompOnFly then
if fWavFileName = '' then
 if fWaveFileList.Count > 0 then
 fWavFileName := fWaveFileList.Strings[0];
if not fCompOnFly then
begin

if WavFileName <> '' then
 nStreams := 2
else
 nStreams := 1;

if fCompStream <> nil then
 AVIStreamRelease(fCompStream);
if Assigned(pfile) then
 repeat
  refcount := AVIFileRelease(pfile);
 until refcount <= 0;

AVIERR := AVIFileOpen(pfile, PChar(TempFileName), OF_READ, nil);

AVIERR := AVIFileGetStream(pfile, fCompStream, streamtypeVIDEO, 0);
if AVIERR <> AVIERR_OK then

AVIERR := CreateEditableStream(fEditStream, fCompStream);
if AVIERR <> AVIERR_OK then

AVIERR := EditStreamSetName(fEditStream, PChar(rsFileSign));
Streams[0] := fEditStream;
if fFourCC = '' then
 CompOptions[0] := nil
else
 CompOptions[0] := @AviCompressoptions;
CompOptions[1] := nil;

AVIERR := AVISaveV(PChar(filename), nil, @AviSaveCallback, nStreams, Streams,
                   CompOptions);
if AVIERR <> AVIERR_OK then
IntToHex(AVIERR, 8));
end;
finally
if fCompStream <> nil then
AVIStreamRelease(fCompStream);
if fEditStream <> nil then
AVIStreamRelease(fEditStream);
```

```
if fPstream <> nil then
AVIStreamRelease(fPstream);
if Stream <> nil then
AVIStreamRelease(Stream);
Stream := nil;
fCompStream := nil;
fPstream := nil;
fEditStream := nil;
fWaveFileList.Clear;
try
repeat
 refcount := AVIFileRelease(pfile);
until refcount <= 0;
pfile := nil;
except
pfile := nil;
end;
if FileExists(TempFileName) then
Deletefile(TempFileName);
fFinalized := false;
end;
end;

const //введення констант
BitCounts: array[pf1Bit..pf32Bit] of byte = (1, 4, 8, 16, 16, 24, 32);

function FourCCToString(f: DWord): TFourCC;
Var // Розділ визначення змінних
S, s1: string;
b: byte;
c: Char;
begin
SetLength(Result, 4);
S := IntToHex(f, 8);
s1 := '$' + copy(S, 7, 2);
b := StrToInt(s1);
c := chr(b);
Result[1] := c;
Result[2] := chr(StrToInt('$' + copy(S, 5, 2)));
Result[3] := chr(StrToInt('$' + copy(S, 3, 2)));
Result[4] := chr(StrToInt('$' + copy(S, 1, 2)));
end;

procedure Tcam_Avi.Compressorlist(const List: TStrings);
Var // Розділ визначення змінних
ii: TICINFO;
i: DWord;
ic: THandle;
BitmapInfoHeader: TBitmapInfoHeader;
Name: WideString;
j: integer;

begin
List.Clear;
List.add(rsNoCompr);

FillChar(BitmapInfoHeader, SizeOf(BitmapInfoHeader), 0);
with BitmapInfoHeader do
begin
biSize := SizeOf(BitmapInfoHeader);
biWidth := fWidth;
biHeight := fHeight;
biPlanes := 1;
biCompression := BI_RGB;
biBitCount := BitCounts[fPixelFormat];
end;

ii.dwSize := SizeOf(ii);
for i := 0 to 200 do
```

```
begin
if ICInfo(ICTYPE_VIDEO, i, @ii) then
begin
ic := ICOpen(ICTYPE_VIDEO, ii.fccHandler, ICMODE_QUERY);
try
 if ic <> 0 then
 begin
  if ICCompressQuery(ic, @BitmapInfoHeader, nil) = 0 then
  begin
   ICGetInfo(ic, @ii, SizeOf(ii));

   Name := '';
   for j := 0 to 15 do
    Name := Name + ii.szName[j];
   List.add(FourCCToString(ii.fccHandler) + ' ' + string(Name));
  end;
 end;
finally
 ICClose(ic);
end;
end;
end;
end;

procedure Tcam_Avi.SetCompression(FourCC: TFourCC);
Var // Розділ визначення змінних
S: string;
ic: THandle;
BitmapInfoHeader: TBitmapInfoHeader;
begin
fFourCC := '';
if FourCC = '' then
exit;
FillChar(BitmapInfoHeader, SizeOf(BitmapInfoHeader), 0);
with BitmapInfoHeader do
begin
biSize := SizeOf(BitmapInfoHeader);
biWidth := fWidth;
biHeight := fHeight;
biPlanes := 1;
biCompression := BI_RGB;
biBitCount := BitCounts[fPixelFormat];
end;
S := FourCC;
ic := ICLocate(ICTYPE_VIDEO, mmioStringToFOURCC(PChar(S), 0), @BitmapInfoHeader,
nil,
ICMODE_COMPRESS);
if ic <> 0 then
begin
fFourCC := FourCC;
ICClose(ic);
end
end;

procedure Tcam_Avi.AddStillImage(const ABmp: TBitmap; Showtime: integer);
Var // Розділ визначення змінних
i: integer;
Samples_Written: LONG;
Bytes_Written: LONG;
AVIERR: HRESULT;
Bitmap: TBitmap;
r1, r2: TRect;
begin
if fAbort then
exit;
if (fFourCC = '') or (not fCompOnFly) then
begin
AddFrame(ABmp);
for i := 1 to (Showtime div FrameTime) do
```

```
begin


AVIERR := AVIStreamWrite(fCompStream, fFrameCount, 1, nil, 0, 0,
Samples_Written, Bytes_Written);

inc(fFrameCount);

if (fFrameCount mod 10 = 0) then
 if Assigned(fOnProgress) then
  fOnProgress(Self, fFrameCount, fAbort);
end;
end
else
begin
Bitmap := TBitmap.Create;
try

Bitmap.PixelFormat := fPixelFormat;


Bitmap.Width := Self.Width;
Bitmap.Height := Self.Height;
Bitmap.Canvas.Lock;
try
 ABmp.Canvas.Lock;
 try
  if fStretch then
   Bitmap.Canvas.stretchdraw
    (Rect(0, 0, Self.Width, Self.Height), ABmp)
  else
   with Bitmap.Canvas do
   begin
    Brush.Color := clBlack;
    Brush.Style := bsSolid;
    FillRect(ClipRect);
    r1 := Rect(0, 0, ABmp.Width, ABmp.Height);
    r2 := r1;
    OffsetRect(r2, (Width - ABmp.Width) div 2, (Height - ABmp.Height) div 2);
    CopyRect(r2, ABmp.Canvas, r1);
   end;
 finally
  ABmp.Canvas.Unlock;
 end;

finally
 Bitmap.Canvas.Unlock;
end;
for i := 0 to (Showtime div FrameTime) do
 InternalAddFrame(Bitmap, true);
finally
Bitmap.Free;
end;
end;
end;

procedure Tcam_Avi.ShowCompressorDialog(ADialogParent: TWinControl);
Var // Розділ визначення змінних
ic: THandle;
S: string;
begin
if fFourCC = '' then
exit;
S := fFourCC;
ic := ICOpen(ICTYPE_VIDEO, mmioStringToFOURCC(PChar(S), 0), ICMODE_QUERY);
try
if ic <> 0 then
begin
```

```
if ICQueryConfigure(ic) then
 ICConfigure(ic, ADialogParent.Handle);
end;
finally
ICClose(ic);
end;
end;

procedure Tcam_Avi.SetPixelFormat(const value: TPixelFormat);
begin
fPixelFormat := value;
end;

procedure Tcam_Avi.InitStreamFormat(const bm: TBitmap);
Var // Розділ визначення змінних
DIB: TDIBSection;
Bits: Pointer;
DIBErr: integer;
S: string;
begin
FillChar(DIB, SizeOf(DIB), 0);
DIBErr := GetObject(bm.Handle, SizeOf(DIB), @DIB);
if DIBErr = 0 then
begin
IntToStr(GetLastError));
end;
if fPInInfo <> nil then
   FreeMem(fPInInfo);
   fPInInfo := nil;
   fInInfoSize := SizeOf(TBitmapInfoHeader);
if DIB.dsbmih.biBitCount <= 8 then
fInInfoSize := fInInfoSize + SizeOf(TRGBQuad) * (1 shl DIB.dsbmih.biBitCount);
   GetMem(fPInInfo, fInInfoSize);
   GetMem(Bits, DIB.dsbmih.biSizeImage);
end;
FillChar(AviCompressoptions, SizeOf(AviCompressoptions), 0);
if fFourCC <> '' then
begin
with AviCompressoptions do
begin
fccType := streamtypeVIDEO;
S := fFourCC;
fccHandler := mmioStringToFOURCC(PChar(S), 0);
dwKeyFrameEvery := round(1000 / fFrameTime);
dwQuality := fCompressionQuality;
lpFormat := fPInInfo;
cbFormat := fInInfoSize;
end;
if fCompOnFly then
begin
end
else
begin
fCompStream := fPstream;
fPstream := nil;
end;
end
else
begin
fCompStream := fPstream;
fPstream := nil;
end;
end;

procedure Tcam_Avi.AddFrame(const ABmp: TBitmap);
Var // Розділ визначення змінних
Bitmap: TBitmap;
r1, r2: TRect;
begin
```

```
if not fCompOnFly or (fFourCC = '') then
begin
AVICreator.AddBitmap(ABmp);
Exit;
end;
if fAbort then
exit;
Bitmap := TBitmap.Create;
try
Bitmap.PixelFormat := fPixelFormat;
Bitmap.Width := Self.Width;
Bitmap.Height := Self.Height;
Bitmap.Canvas.Lock;
try
ABmp.Canvas.Lock;
try
 if fStretch then
  Bitmap.Canvas.StretchDraw(Rect(0, 0, Self.Width, Self.Height), ABmp)
 else
  with Bitmap.Canvas do
  begin
   Brush.Color := clBlack;
   Brush.Style := bsSolid;
   FillRect(ClipRect);
   r1 := Rect(0, 0, ABmp.Width, ABmp.Height);
   r2 := r1;
   OffsetRect(r2, (Width - ABmp.Width) div 2, (Height - ABmp.Height) div 2);
   CopyRect(r2, ABmp.Canvas, r1);
  end;
finally
 ABmp.Canvas.Unlock;
end;

finally
Bitmap.Canvas.Unlock;
end;

InternalAddFrame(Bitmap, true);

finally
Bitmap.Free;
end;
end;

procedure Tcam_Avi.SetHeight(const Value: integer);
begin
fHeight := Value and ($FFFFFFFF - 15);
end;

procedure Tcam_Avi.SetWidth(const Value: integer);
begin
fWidth := Value and ($FFFFFFFF - 15);
end;

initialization // Ініціалізація модуля
OwnerForm := nil;
end.
```