Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня
магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Самарця Назара Олександровича*
	(ПІБ)
академічної групи	*121М-20-1*
	(шифр)
спеціальності	*121 Інженерія програмного забезпечення*
	(код і назва спеціальності)
освітньої програми	*«Інженерія програмного забезпечення»*
	(назва освітньої програми)
на тему:	*Розробка програмного забезпечення аналізу особистості на основі зображення відбитків*

_____*Н.О. Самарець*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
розділів кваліфікаційної роботи				
спеціальний	*Проф. Якунін А.О.*			
економічний	*Проф. Вагонова О.Г.*			
Рецензент	*Проф. Байбуз О.Г.*			
Нормоконтролер	*Доц. Приходченко С.Д.*			

Дніпро
2022

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 21 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121м-20-1* *Самарцю Назару Олександровичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Розробка програмного забезпечення аналізу особистості на основі зображення відбитків*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від ___.___.2021 р. № _____

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – методи аналізу особистості на основі зображення відбитків.

Предмет досліджень – програмне забезпечення аналізу особистості на основі зображення відбитків.

Методи дослідження: методи пошук мінюцій, фільтрації збурень, розробка алгоритмів та програмного забезпечення аналізу відбитків.

Мета роботи – дослідження методів та можливості їх застосування для покращення якості розпізнавання зображень відбитків.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень визначається тим, що розроблено новий оригінальний алгоритм для попередньої обробки та розпізнавання відбитків.

Практична цінність результатів полягає у тому, що в результаті проведеного дослідження було спроектовано алгоритм покращення якості розпізнавання зображень відбитків.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Система призначена для обробки растрових зображень. Внаслідок неточностей, шумів і апроксимацій, що вносяться обладнанням (сканер або будь-яке інший пристрій, що дискретизирує графіку) в зображенні з'являються шуми різної природи. Система дозволяє частково позбутися від цих спотворень. Тому якість вхідних образів має бути на прийнятному рівні.

Основним видом інформації, що обробляється в системі, є графічна інформація в растровому поданні та її об'єктне уявлення. Такий вид даних сприймається людиною безпосередньо і цілісно, тому необхідно забезпечити засоби наглядної візуалізації зображень на різних етапах обробки.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз джерел та постановка задачі	12.09.2021 - 30.09.2021
Побудова математичних моделей та розробка алгоритму	01.10.2021 - 31.10.2021
Розробка та тестування програмного забезпечення для розпізнавання відбитків	01.11.2021 - 30.12.2021

Завдання видав _____ *Якунін А.О.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Самарець Н.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *12.09.2021 р.*

Термін подання кваліфікаційної роботи до ЕК *20.01.2021*

РЕФЕРАТ

Пояснювальна записка: ___ стор., ___ рис., ___ таблиці, ___ додатка, ___ джерел.

Об’єкт досліджень – методи аналізу особистості на основі зображення відбитків.

Предмет досліджень – програмне забезпечення аналізу особистості на основі зображення відбитків.

Методи дослідження: методи пошук мінюцій, фільтрації збурень, розробка алгоритмів та програмного забезпечення аналізу відбитків.

Мета роботи – дослідження методів та можливості їх застосування для покращення якості розпізнавання зображень відбитків.

Новизна запропонованих рішень визначається тим, що розроблено новий оригінальний алгоритм для попередньої обробки та розпізнавання відбитків.

Практична цінність результатів полягає у тому, що в результаті проведеного дослідження було спроектовано алгоритм покращення якості розпізнавання зображень відбитків.

Система призначена для обробки растрових зображень. Внаслідок неточностей, шумів і апроксимацій, що вносяться обладнанням (сканер або будь-яке інший пристрій) в зображенні з'являються шуми різної природи. Система дозволяє частково позбутися від цих спотворень. Тому якість вхідних образів має бути на прийнятному рівні.

Основним видом інформації, що обробляється в системі, є графічна інформація в растровому поданні та її об'єктне уявлення. Такий вид даних сприймається людиною безпосередньо і цілісно, тому необхідно забезпечити засоби наглядної візуалізації зображень на різних етапах обробки.

У розділі «Економіка» проведені розрахунки трудомісткості розробки програмного забезпечення, витрат на створення ПЗ і тривалості його розробки, а також проведені маркетингові дослідження ринку збуту створеного програмного продукту.

Список ключових слів: РОЗПІЗНАВАННЯ ВІДБИТКІВ, ЯКІСТЬ РОЗПІЗНАВАННЯ, ОБРОБКА СИГНАЛУ, ФІЛЬТРАЦІЯ ЗБУРЕНЬ, ПОШУК МІНЮЦІЙ.

ABSTRACT

Explanatory note: ___ pages, ___ figures, ___ tables, ___ appendices, ___ sources.

The object of research is methods of personality analysis based on the image of prints.

The subject of research is personality analysis software based on fingerprint images.

Research methods: methods of miniature search, perturbation filtering, development of algorithms and software for fingerprint analysis.

The aim of the work is to study the methods and possibilities of their application to improve the quality of image recognition.

The novelty of the proposed solutions is determined by the fact that a new original algorithm for pre-processing and fingerprint recognition has been developed.

The practical value of the results is that as a result of the study, an algorithm was designed to improve the quality of image recognition.

The system is designed for processing raster images. Due to inaccuracies, noise and approximations made by the equipment (scanner or any other device) in the image there are noises of different nature. The system allows you to partially get rid of these distortions. Therefore, the quality of the input images must be at an acceptable level.

The main type of information processed in the system is graphic information in raster representation and its object representation. This type of data is perceived by a person directly and holistically, so it is necessary to provide a means of visual visualization of images at different stages of processing.

In the section "Economics" calculations of the complexity of software development, the cost of creating software and the duration of its development, as well as marketing research of the market for the software product.

List of keywords: IMPRINT RECOGNITION, QUALITY RECOGNITION, SIGNAL PROCESSING, DISTURBANCE FILTRATION, MINUTE SEARCH.

ЗМІСТ

ВСТУП

У наш час паролі, персональні ідентифікаційні номери і спеціальні ідентифікаційні картки стали життєвою необхідністю. Наприклад, щоб отримати готівку з банкомату, Вам потрібно код PIN; щоб отримати доступ до поштовій програмі або до певної категорії комп'ютерних даних, необхідний пароль. У світлі останніх подій, що відбуваються в світі, особливо в зв'язку з ростом активності міжнародного тероризму, питанням безпеки приділяється все більше уваги.

Таким чином, людина повинна зберігати в своїй пам'яті безліч різних комбінацій цифр і букв. Щоб полегшити долю сучасної людини, компанії, що спеціалізуються на виробництві комп'ютерів, почали займатися розробкою біометричних технологій. Біометрія – ця наука, що вивчає можливості використання різних характеристик людського тіла (будь то відбитки пальців або унікальні властивості людського зіниці або голосу) для ідентифікації кожної конкретної людини. Користуючись біометричними технологіями, людина ніколи не зможе забути необхідний йому пароль або код, оскільки його великий палець, голос або зіницю ока завжди знаходяться з ним [1].

Відбиток пальця утворює так звані папілярні лінії на гребішкових виступах шкіри, розділених борозенками. З цих ліній складаються складні візерунки (дугові, петльові та завітковие), які мають властивості індивідуальності й неповторності, що дозволяє абсолютно надійно ідентифікувати особу. Хоча відсоток відмови в доступі уповноважених користувачів становить близько 3%, відсоток помилкового доступу – менше одного до мільйона. Переваги доступу по відбитку пальця – простота використання, зручність і надійність. Весь процес ідентифікації займає мало часу і не вимагає зусиль від тих, хто використовує дану систему доступу. Дослідження також показали, що використання відбитка пальця для ідентифікації особи є найбільш зручним з усіх біометричних методів. Імовірність помилки при ідентифікації користувача набагато менше в порівнянні

з іншими біометричними методами [2]. Крім того, пристрій ідентифікації за відбитками пальців не вимагає багато місця на клавіатурі або в механізмі.

Отриманий образ відбитку пальця – це растр, який можна описати особливим чином, грунтуючись на будові папілярного візерунка. Виявивши структуру відбитка його можна порівняти з іншими відбитками і виявити ті, які є аналогічними або ж сказати, що відбитки різні.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМИ ІДЕНТИФІКАЦІЇ ВІДБИТКУ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1. Технології розпізнавання особистості за візерунком відбитків

1.1.1. Система управління BioLink BioTime

BioTime – це сучасна біометрична система обліку робочого часу і контролю фізичного доступу. BioTime інтегрує і ефективно реалізує ключові функції управління персоналом (облік, аналіз і контроль використання робочого часу) і безпеки (розмежування доступу в будівлі і приміщення).

Достовірність звітів і надійність захисту забезпечується ідентифікацією співробітників по унікальним біометричними параметрами - відбитками пальців. Реєстрація парафій і відходів співробітників здійснюється за пред'явленням ідентифікаторів автоматично – з оптимальною швидкістю та незмінною точністю.

З системою BioTime працювати зручно і легко. Дружній, продуманий інтерфейс можна швидко налаштовувати під конкретні завдання, велика номенклатура звітів задовольняє максимум потреб керівників, працівників кадрових служб і відділів безпеки, бухгалтерів і менеджерів оперативного і тактичного ланки. Інтеграція BioTime в корпоративну інформаційну систему також здійснюється вільно і плавно, і не випадково керівники IT-служби і системні адміністратори одними з перших підтримують впровадження BioTime.

Система BioTime адаптована під потреби підприємств, що діють в різних масштабах.

Директори компаній малого і середнього бізнесу гідно оцінюють ефективність і економічність BioTime, простоту експлуатації цієї системи, розумні системні вимоги, які виконуються навіть в самому невеликому офісі.

Керівників великих підприємств приваблює можливість застосовувати BioTime в мережі територіально розподілених філій і відділень - при тому, що повністю забезпечені централізоване управління системою і можливості її масштабування при успішному розвитку і подальшому зростанні компанії. BioTime враховує управлінську ієрархію, дозволяє налаштовувати права і повноваження менеджерів, чітко регламентуючи можливості доступу кожного з них до необхідної інформації.

Безперервно нарощується функціонал системи BioTime в частині контролю фізичного доступу. Впроваджувати біометричні технології для вирішення цих завдань можна і з «чистого аркуша» (при в'їзді в новий офіс чи зміні робочого приміщення), і в уже існуючі системи контролю доступу (доповнюючи, наприклад, наявні карткові зчитувачі ідентифікацією за відбитками пальців при доступі в особливо важливі приміщення).

Біометрія визнана експертами найперспективнішою і швидко розвивається технологією контролю доступу, яку використовують в найбільших банках, державних установах, на підприємствах і виробництвах особливої важливості (аеропорти, вокзали, порти, електростанції, машинобудівна, хімічна, оборонна промисловість, підприємства аерокосмічної галузі та інші .) [5].

1.1.2. Біометрична система BioLink IDenium

BioLink IDenium – єдина сертифікована біометрична система аутентифікації користувачів, що дозволяє повністю замінити використання паролів на біометричну або многофакторную аутентифікацію. Впровадження IDenium забезпечує надійний захист доступу до інформаційних ресурсів організації, а також дозволяє ефективно мінімізувати ризики несанкціонованого доступу і інсайдерства.

Система BioLink IDenium дозволяє встановити біометричну аутентифікацію при доступі в операційну систему, корпоративні програми та інформаційні системи, а також веб-сайти. Завдяки IDenium, Ви так само можете встановити аутентифікації за відбитками пальців при проведенні критично важливих операцій (переказ коштів, доступ до певної інформаційних ресурсів, видалення файлів).

Для посиленого захисту доступу система BioLink IDenium дозволяє використовувати багатофакторну аутентифікацію «смарт-карта + відбиток», завдяки сканеру відбитків пальців BioLink U-Match 5.0 з інтегрованим зчитувачем смарт-карт. При використанні даної конфігурації pin-код, прив'язаний до смарт-карті, замінюється на відбиток пальця користувача.

Технологія єдиного доступу (Single Sign-On) до всіх корпоративних додатків забезпечує не тільки зручність для користувачів і мінімізацію ризиків витоку паролів, але і значно знижує навантаження на служби Help Desk по відновленню загублених паролів.

IDenium зберігає повну історію подій аутентифікації, завдяки чому ви завжди можете дізнатися, який співробітник і в який час отримував доступ до певного додатка. Налаштування прав доступу до ярку подій дозволяють строго обмежити доступ до цієї інформації, наприклад, лог може бути доступний тільки співробітникам служби ІБ.

Завдяки комплекту розробника IDenium SDK, Ви можете інтегрувати процес біометричної аутентифікації в будь-які корпоративні інформаційні системи з урахуванням індивідуальної бізнес-логіки.

Система IDenium підтримує необмежене число додаткових серверів, що гарантує рівномірний розподіл навантаження на сервери біометричних даних, забезпечення відмовостійкості і швидкої реакції під час пікових навантажень.

Дослідження Gartner Group показало, витрати кожної компанії на адміністрування паролів складають від 150 до 220 USD на одного користувача

щорічно. Це і неефективне використання робочого часу співробітників через часті і тривалих звернень в службу підтримки і збільшення навантаження з боку системних адміністраторів.

Згідно даній статистиці, впровадження біометричної системи IDenium дозволяє заощадити компанії зі штатом 1000 чоловік до 180 000 USD рік. Це, без сумніву, серйозні гроші для будь-якої компанії [5].

1.2. Обґрунтування доцільності розробки системи ідентифікації особи за відбитками пальців

Завдання структурного аналізу зображень мають широкий спектр застосування, починаючи від векторизації растрових і закінчуючи розпізнаванням образів. Структурний аналіз зображень має на увазі виділення з них структурних елементів, таких, наприклад, як лінія, область, компактний елемент (буква) і так далі.

На даний момент надійна інформаційна захист є одним з основних критеріїв, за якими повинні відбиратися системи, призначені для зберігання і обробки важливої інформації. Це обумовлено існуючої ймовірністю несанкціонованого доступу в такі системи, оскільки вони мають широке інформаційну взаємодію із суміжними системами управління через мережу internet. Тому забезпечення інформаційної безпеки має бути найважливішим етапом при їх розробці [3].

Захист на основі біометричних параметрів людського тіла, зокрема по відбитку пальця, має низку незаперечних полюсів: простота використання, зручність і надійність. Весь процес ідентифікації займає мало часу і не вимагає зусиль від тих, хто використовує дану систему доступу. Дослідження також показали, що використання відбитка пальця для ідентифікації особи є найбільш зручним з усіх біометричних методів. Імовірність помилки при ідентифікації

користувача набагато менше в порівнянні з іншими біометричними методами. Крім того, пристрій ідентифікації за відбитками пальців не вимагає багато місця на клавіатурі або в механізмі.

У більшості випадків робота з важливою інформацією на увазі також своєчасне прийняття рішень і безперервне управління ходом виконання. У зв'язку з цим існує необхідність безперервного підтвердження особи (в разі якщо людина з якоїсь причини покине своє робоче місце, то будь-який в цей час зможе задавати команди телеуправління або відповідальні команди). Таке підтвердження особи метод «єдиного входу в мережу» надати не може, а вводити пароль після кожної команди обтяжливо [4].

Хоча на ринку існують готові системи, але на ряду зі своїми перевагами вони мають ряд недоліків, таких як закритість вихідного коду і алгоритму, як наслідок неможливість застосування в своїх системах, а також висока ціна. Внаслідок чого є сенс в розробці системи, яка б надавала можливість всім розробникам мати готову базу для розробки власних проектів на основі біометричних технологій. А також надати об'єктне опис різних, не тільки папілярного візерунка, зображень.

Створювана система носить пошуково-дослідницький характер і спрямована на полегшення розробки алгоритмів обробки зображень, спрощення аналізу експериментальних даних і виявлення загальних закономірностей.

Система ідентифікації особи за відбитками пальців реалізує визначення особистості на основі біометричних параметрів людського тіла, а саме побудова відбитків пальців. Система призначена для обробки графічних зображень відбитків. Система дозволяє порівняти кілька відбитків один з одним по виділеним локальним особливостям. Локальними особливостями є мінюції і їх відносні параметри (розташування одних мінюцій щодо всіх інших), що гарантує незалежність порівняння від паралельного перенесення і обертання.

Програмний продукт знайде застосування в різних прикладних системах [3], включаючи:

1) системи цивільного ідентифікації;

2) криміналістичні системи ідентифікації;

3) великомасштабні комерційні додатки.

Системи цивільного ідентифікації включають в себе:

– водійські паспорта;
– національні ідентифікаційні карти громадян;
– реєстрація виборців;
– реєстрація для соціальних програм;
– імміграційна реєстрація, візи;
– ідентифікація співробітників державних установ.

Криміналістичні системи ідентифікації включають:

– чи знаходиться даний громадянин в розшуку?;
– колишні судимості;
– реєстрація укладених / контроль доступу;
– мобільні та віддалені програми;
– обробка слідів відбитків пальців, отриманих з місць злочину.

Великомасштабні комерційні додатки включають:

– доступ до web-ресурсів, електронна комерція;
– доступ для користувачів і співробітників;
– фінансові послуги, перевірка оплати;
– доступ в будівлі і приміщення;
– програми лояльності.

1.3. Постановка задачі дослідження.

На основі аналізу були сформовані основні цілі створення підсистеми, функціональне призначення, особливості та її вимоги.

Дослідження показали, що використання відбитка пальця для ідентифікації особи є найбільш зручним з усіх біометричних методів. Імовірність помилки при ідентифікації користувача набагато менше в порівнянні з іншими біометричними методами [2]. Крім того, пристрій ідентифікації за відбитками пальців не вимагає багато місця на клавіатурі або в механізмі.

Створення підсистеми графічної обробки зображення відбитків пальців дозволить отримати нові можливості в сфері захисту інформації, створенню нових ефективних алгоритмів по обробці растра та .

Підсистема використовує цифрове відскановане зображення.

Реалізація системи ідентифікації особи за відбитками дозволить інтегрувати в єдиному інтерфейсі всі етапи обробки зображення відбитка пальця і порівняння його з іншими відбитками:

1) модифікація зображення, виправлення викривлень;

2) виділення локальних особливостей – мінюцій. Формування списку мінюцій в абсолютних параметрах;

3) сортування списку абсолютних параметрів, виключення помилкових і ненадійних мінюцій;

4) конвертація абсолютних параметрів у відносні, формування списку відносних параметрів;

5) установка системи допусків для обліку кореляції зображень.

Також підсистема аналізу зображення повинна забезпечувати можливість отримання основних статистичних характеристик папілярного візерунка за ключовими ділянках.

Підсистема порівняння зображень відбитків служить для автоматизованого виявлення схожості різних зображень папілярного візерунка.

РОЗДІЛ 2
РОЗРОБКА МЕТОДИКИ ІДЕНТИФІКАЦІЇ ОСОБИ НА ОСНОВІ ПАПІЛЯРНОГО ВІЗЕРУНКА

2.1. Моделі представлення зображень

Графічні образи, які подаються навколишнім світом людині, володіють великим розмаїттям. Невід'ємним атрибутом зображення є його просторова структура. Здатність реконструювати цю структуру при візуальному сприйнятті і забезпечує предметність сприйняття.

Просторова визначеність, яка полягає в тому, що будь-яка точка зображення належить єдиному і цілком певному структурному елементу, який може представляти об'єкт або належить одному або кільком об'єктам відповідної предметної області. Таким чином, будь-яке зображення будується відповідно до деякого апріорним планом, який визначає місце положення і смислові характеристики його структурних елементів.

Моделі представлення зображень в ЕОМ можна розділити на два типи: растрові і векторні.

Найбільш поширена форма подання «сирих» зображень на ЕОМ - це растр. Зображення в цьому випадку представляє собою матрицю з NxM точок (пікселів). Візуалізація растрових зображень досить проста і полягає в порядковому виведенні його пікселів на екран. Однак модель цього типу не несе в собі структурної і тим більше семантичної інформації, що обмежує сферу її застосування. При введенні зображень з реального світу в ЕОМ вони часто постають у растрової формі.

Векторні моделі подання зображень засновані на тому, що будь-яку лінію можна уявити в аналітичному вигляді, наприклад у вигляді сукупності векторів - спрямованих відрізків. Візуалізація зображень в векторної моделі складніше, ніж

в растровій. Але модель набуває когнітивної за рахунок включення в неї структурної інформації.

Образ відбитка пальця, як правило, зберігається в двійковому коді, де кожен піксель малюнка описується 8 бітами, тобто 256 відтінками сірого кольору. У передових системах сканування цифровий образ відбитку обробляється за допомогою спеціального алгоритму покращення зображення. Цей алгоритм забезпечує зворотний зв'язок з датчиком для регулювання параметрів сканування. Коли датчик фіксує остаточний образ, алгоритм налаштовує контрастність і чіткість зображення відбитка для отримання найкращої якості [12].

Методи розпізнання відбитку пальця засновані на порівнянні зі зразками або на використанні характерних деталей.

При розпізнанні по деталях з образу витягуються тільки специфічні місця, де знайдена особливість (деталь). Зазвичай це або закінчення гребеня, або його роздвоєння (рисунок 2.1). Зміст шаблону в цьому випадку становлять відносні координати і відомості про орієнтацію деталі. Розпізнає алгоритм відшукує і порівнює між собою відповідні деталі. Ні поворот відбитка пальця, ні його паралельне перенесення не впливають на функціонування системи, оскільки алгоритм працює з відносними величинами.

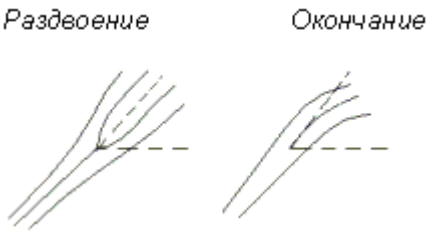

Рис. 2.1 Типи мінюцій

Для порівняння на бітовому образі проводиться пошук локальних особливостей папілярного візерунка – мінюцій. Для пошуку використовується

алгоритм обходу по контуру гребенів. В результаті підсистема аналізу реалізує перехід від растрового представлення до структурної поданням.

Вхідна інформація

Вхідний інформацією є бітовий растр відбитка, отриманий за допомогою сканування дозволом 600dpi. Розширення бітового файлу по-замовчуванню *.bmp. Формат bmp (від слів BitMaP - бітова масив) представляє з себе нестиснене (в основному), що дозволяє не вносити похибок, зображення, яке досить легко читається і виводиться в ОС Windows, в якій є спеціальні функції API, які в цьому допомагають [13].

Вихідна інформація

Вихідною інформацією є список мінюцій в абсолютних параметрах, розташований в пам'яті, що містить параметри кожної знайденої мінюції. Кожен елемент масиву містить всі необхідні параметри мінюції: координати цілого типу – 2х4 байта, кут напрямку 8 байт, тип точки 1 байт.

Структура масиву:

$$M = \begin{vmatrix} X_1 & Y_1 & \alpha_1 & T_1 \\ \dots & & & \\ X_k & Y_k & \alpha_k & T_1 \end{vmatrix}$$

Таблиця 2.1

Формат рядка файлу зі структурним описом

Поле	Формат	Опис
X	Ціле	Абсциса мінуцій на растрі
Y	Ціле	Ордината мінуцій на растрі
α	Ціле	Орієнтація мінуцій на растрі
T	Байт	Тип мінюції (роздвоєння чи закінчення)
k	Ціле	Кількість мінюцій

2.2. Математична постановка задачі усунення дефектів

Робота підсистеми реалізується наступними етапами:

– коригування вхідного образу, усунення дефектів і спотворень;

– пошук мінюцій і формування списку їх абсолютних параметрів;

– фільтрація отриманого списку параметрів;

Для вирішення поставлених завдань потрібні стандартні операції для роботи з масивом, які представлені у таблиці 2.2

Таблиця 2.2

Операції над масивом

Позначення	Розшифрування
|Array|	кількість елементів масиву
Array[i]	звернення до і-му елементу масиву
$E \subset M$	операція додавання елемента E в кінець масиву M
Delete(Array,Pos)	операція видалення елемента на позиції Pos з масиву Array
Delete(Array, Element)	операція видалення елемента Element з масиву Array
Pos(Arr--ay,Element)	операція отримання номера елемента Element в масиві Array
Array1∪Array2	операція додавання в кінець масиву Array1 не дубльованих елементів масиву Array2
Array1∩Array2	операція перетину масивів
$E \in M$	логічна операція приналежності елемента E масиву M
Sort($Array_f$)	операція сортування масиву Array за значенням поля f

Папілярний візерунок представлений у вигляді растра містить окремі елементи лінії. Лінії – це відображення гребенів папілярного візерунка, обхід по контуру цих ліній дозволить виділити окремі гребені і отримати інформацію про їхнє розташування на пальці. Однак в процесі отримання растру відбитка можливі типові дефекти зображення, які виникають внаслідок попадання сміття на скануючий пристрій, існування порізів і складок на шкірі, мінливому притиску пальця до сканера і зміна його положення при кожному новому скануванні.

На рисунку 2.2 показаний розрив лінії, при цьому виконується така умова:

$$\sqrt{(\mathrm{A}.x - B.x)^2 + (\mathrm{A}.y - B.y)^2} < \Delta) \wedge (C < 150°) \wedge (D_1 \cap L_1 \neq \emptyset) \wedge (D_2 \cap L_2 \neq \emptyset) \quad (2.1)$$

де A = {x, y};

B = {x, y};

Δ - емпірична величина.

На рисинку 2.2 показано злипання ліній, при цьому виконується умова 2.1.

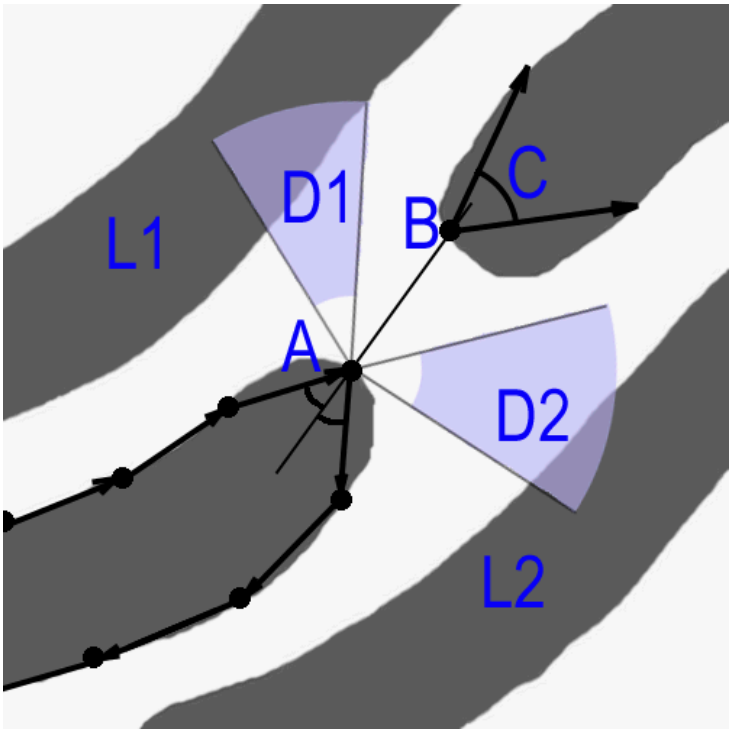

Рис. 2.2 – Розрив лінії

A – сильне викривлення контуру лінії папілярного візерунка;

B – ймовірна точка постійного розвитку папілярного візерунка;

C – викривлення контуру у ймовірній точці продовження;

D1, D2 – прилеглі області;

L1, L2 – ймовірні сусідні лінії папілярного візерунка.

Дефекти бувають двох видів – це злипання сусідніх гребенів і обриви гребеня на растрі внаслідок описаних вище ситуацій. Так як дані дефекти частково передбачувані, то можна їх усунути. Застосовуючи підготовку зображення, до подальшого структурному аналізу, вдається значно знизити кількість шумів і спотворень у вихідному растрі, що веде до підвищення швидкості і надійності розпізнавання.

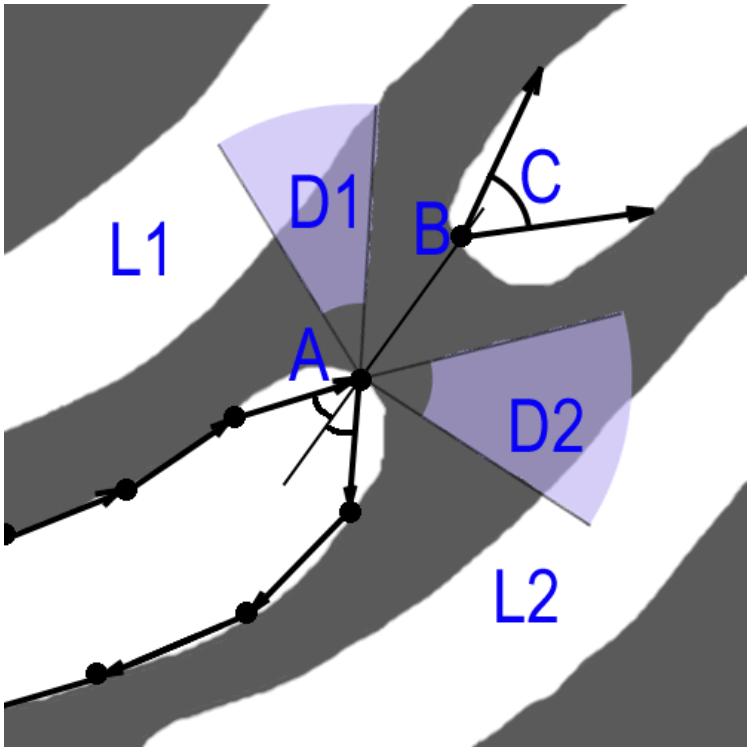

Рис. 2.3 Злипання ліній

A – сильне викривлення контуру лінії папілярного візерунка;

B – ймовірна точка постійного розвитку папілярного візерунка;

C – викривлення контуру у ймовірній точці продовження;

D1, D2 – прилеглі області;

L1, L2 – ймовірні сусідні западини папілярного візерунка.

В результаті рішення задачі виявлення і усунення дефектів сканування система ідентифікації особистості доповнить свої функціональні можливості здатністю підвищення якості вхідних образів.

Пошук мінюцій відбувається по знаходженню локальних особливостей. Локальні особливості – це сильні викривлення контуру ліній. Викривлення, які є мінюціями – це закінчення і роздвоєння, але крім них існують злипання сусідніх ліній і обриви одній лінії, що не є мінюціями.

2.3. Алгоритм вирішення задачі усунення дефектів

Виділимо основні дії по обробці кожної лінії на зображенні відбитка:

1. Виділити довільну чорну крапку на растрі, що належить оброблюваної лінії, і зробити обхід по контуру лінії папілярного візерунка, якій належить ця точка.

2. Якщо виявлено ділянку розриву, то виконується відновлення цілісності лінії;

3. Якщо виявлено ділянку злипання, то виконується роз'єднання ліній.

Результатом роботи є растр більш придатний для пошуку на ньому мінюцій, ніж початковий.

R - Бітовий растр

Map - список. Map = {x, y} і

R.GetPixelColor (x, y) – отримати значення кольору пікселя з координатами {x, y} на растрі R

R.FloodFill (x, y, color) – залити область з кольором R.GetPixelColor (x, y) в колір color

R.width () – ширина растра в пікселях

R.height () – висота растра в пікселях

R.ChangeLine (Map [i]) – обхід по контуру лінії з точки Map [i]

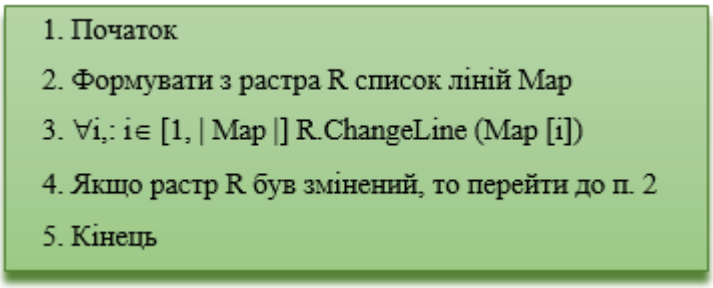

Рис. 2.4 Алгоритм розв'язання задачі

Опис алгоритму «Формування списку ліній».

Алгоритм для знаходження на растрі точок належать різним папілярним лініях.

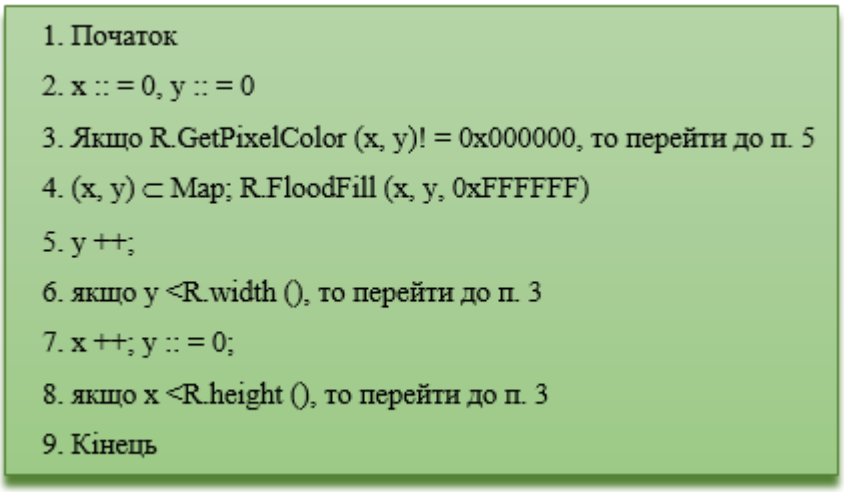

Рис. 2.5. Алгоритм «Формування списку ліній»

Опис алгоритму «ChangeLine».

Алгоритм для пошуку злипання, обривів і усунення їх на растрі.

dot0, dot1 – точка належать контуру лінії

vec0, vec1 – локальні напрямки

GetVec (dot0, dot1) – напрям з точки dot0 в dot1

alphaTest – зумовлена константа визначає сильне викривлення контуру папілярної лінії

NextDotCW (dot0, step) – отримання координат точки наступної через step точок

Умови обриву і злипання описані в вище.

1. Початок
2. dot0 :: = початкове значення
3. dot1 :: = NextDotCW (dot0, step);
4. vec0 :: = GetVec (dot0, dot1);
5. dot0 :: = dot1;
6. dot1 :: = NextDotCW (dot0, step);
7. vec1 :: = GetVec (dot0, dot1);
8. Якщо | vec1 - vec0 | <AlphaTest, то перейти до п. 11
9. Якщо знайдена точка є злипання, то роз'єднати лінії
10. Якщо знайдена точка є обривом, то відновити цілісність лінії
11. Якщо обхід по контуру привів до початкової точки, то перейти до п.13
12. vec0 :: = vec1; перейти до п.5
13. Кінець

Рис. 2.6. Алгоритм «ChangeLine»

2.4. Математична постановка задачі пошуку мінюцій

Пошук мінюцій відбувається по знаходженню локальних особливостей. Локальні особливості це сильні викривлення контуру ліній, одні з викривлень є мінюціями – це закінчення і роздвоєння, але крім них існують злипання сусідніх ліній і обриви одній лінії.

На рисунку 2.7 показано «закінчення», при цьому виконується наступна умова:

$$\sqrt{(\mathrm{A}.x - B.x)^2 + (\mathrm{A}.y - B.y)^2} > \Delta) \wedge (C \geq 150°) \wedge (D_1 \cap L_1 \neq \emptyset) \wedge (D_2 \cap L_2 \neq \emptyset) \quad (2.2)$$

де A = {x, y};

B = {x, y};

Δ – емпірична величина.

На рисунку 2.7 показано «роздвоєння», при цьому виконується умова 2.2.

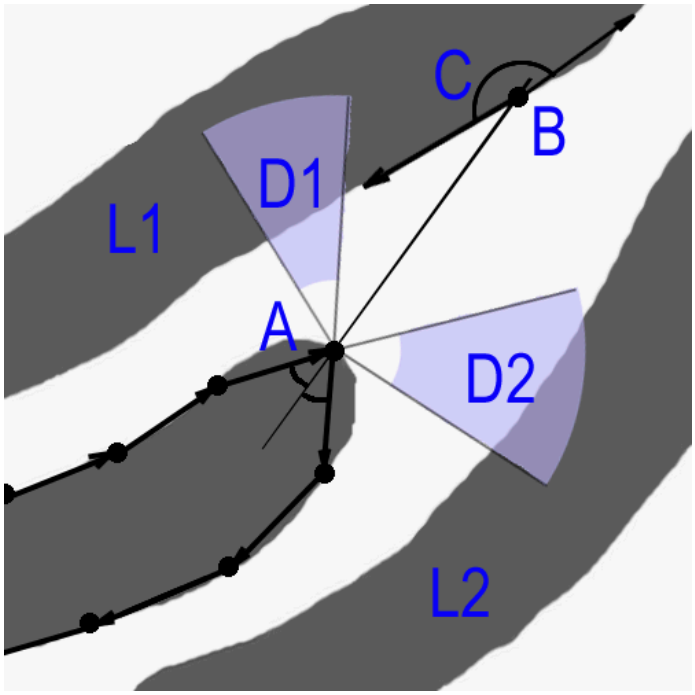

Рис. 2.7. «Закінчення»

A – сильне викривлення контуру лінії папілярного візерунка;

B – ймовірна точка постійного розвитку папілярного візерунка;

C – викривлення контуру у ймовірній точці продовження;

D1, D2 – прилеглі області;

L1, L2 – ймовірні сусідні лінії папілярного візерунка.

Вхідною інформацією є бітовий растр після попередньої обробки. Растр має глибину 1 біт на піксель і дозвіл 600dpi. Формат bmp (від слів BitMaP - бітова карта) представляє з себе нестиснене (в основному), що дозволяє не вносити похибок, зображення. Формат bmp досить легко читається і виводиться в ОС Windows, в якій є спеціальні функції API [11].

Вихідною інформацією є список параметрів, де були виявлені специфічні точки (особливість, деталь), в абсолютних параметрах.

Список, розташований в пам'яті, на даному етапі містить крім потрібних точок – помилкові, які утворюються при неякісному вхідному образі. Кожен елемент масиву містить всі необхідні параметри: координати цілого типу – 2x4 байта, кут напрямку 8 байт, тип точки 1 байт.

Структура масиву:

$$M = \begin{vmatrix} X_1 & Y_1 & \alpha_1 & T_1 \\ \dots & & & \\ X_k & Y_k & \alpha_k & T_1 \end{vmatrix}$$

Xi, Yi – Координати мінюції на растрі

αi – Орієнтація мінюції

T – Тип (закінчення або роздвоєння)

k – Кількість мінюцій

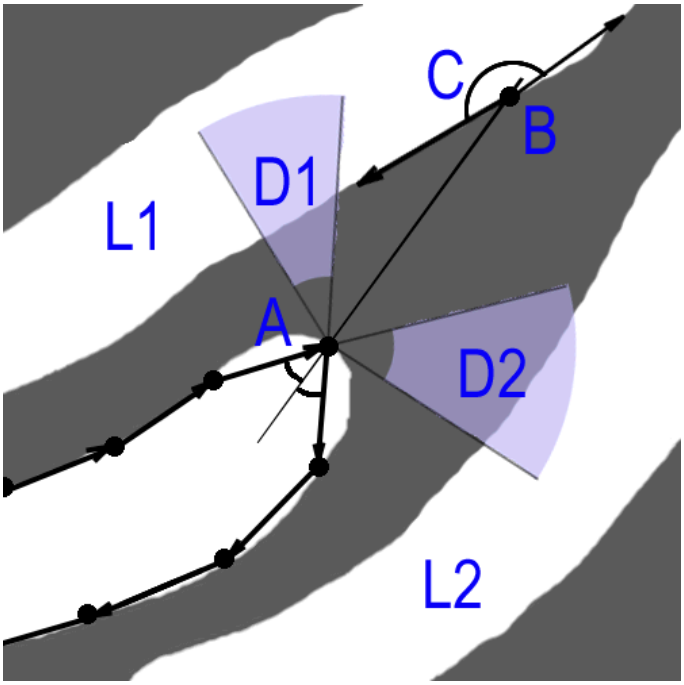

Рис. 2.8. «Роздвоєння»

A – сильне викривлення контуру лінії папілярного візерунка;

B – ймовірна точка постійного розвитку папілярного візерунка;

C – викривлення контуру у ймовірній точці продовження;

D1, D2 – прилеглі області;

L1, L2 – ймовірні сусідні западини папілярного візерунка.

2.5. Алгоритм розв'язання задачі пошуку мінюцій

Виділимо основні дії по обробці кожної лінії на зображенні відбитка:

Виділити довільну чорну крапку на растрі, що належить оброблюваної лінії, і зробити обхід по контуру лінії папілярного візерунка, якій належить ця точка;

Якщо виявлено мінюція, то запишемо її координати в список.

Результатом є список параметрів, з виявленими специфічними точками (особливість, деталь), в абсолютних параметрах. Список на даному етапі містить крім потрібних точок - помилкові, які утворюються при неякісному вхідному образі.

R – бітовий растр

Map – список. Map = {x, y} i

R.GetPixelColor (x, y) – отримати значення кольору пікселя з координатами {x, y} на растрі R

R.FloodFill (x, y, color) – залити область з кольором R.GetPixelColor (x, y) в колір color

R.width () – ширина растра в пікселах

R.height () – висота растра в пікселах

R.ReadLine (Map [i]) – обхід по контуру лінії з точки Map [i], отримує список координат мінюцій.

1. Початок
2. Формувати з растра R список ліній Map
3. ∀i,: i∈ [1, | Map |] List :: = R.ReadLine (Map [i])
4. Висновок List
5. Кінець

Рис. 2.9. Алгоритм розв'язання задачі

Опис алгоритму «ReadLine»

Алгоритм для пошуку закінчень і роздвоєнь, формування списку параметрів локальних особливостей.

dot0, dot1 – точки належать контуру лінії

vec0, vec1 – локольние напрямки

GetVec (dot0, dot1) – напрям з точки dot0 в dot1

alphaTest – зумовлена константа

NextDotCW (dot0, step) – отримання координат точки наступної через step точок

Return – повертається список

Умови обриву і злипання описані в вище.

1. Початок
2. dot0 :: = початкове значення
3. dot1 :: = NextDotCW (dot0, step);
4. vec0 :: = GetVec (dot0, dot1);
5. dot0 :: = dot1;
6. dot1 :: = NextDotCW (dot0, step);
7. vec1 :: = GetVec (dot0, dot1);
8. Якщо | vec1 - vec0 | <AlphaTest, то перейти до п. 11
9. type :: = vec1 <vec0;

alpha :: = можливе напрям продовження лінії;

10. {dot0, alpha, type} ⊂ Return
11. Якщо обхід по контуру привів до початкової точки, то перейти до п.13
12. vec0 :: = vec1; перейти до п.5
13. Кінець

Рис. 2.10. Алгоритм «ReadLine»

В результаті виділення спеціальних точок, є такі, які не є мінюціями і можуть не бути присутнім при наступному аналізі, що негативно вплине на результат порівняння і швидкість роботи, так як розмір оброблюваної інформації буде більше. Для виключення таких точок виведемо правила надійної точки:

– пара точок не може перебувати ближче певної відстані;
– пара точок мають однаковий тип і спрямовані один на одного не можуть перебувати ближче 3 * d, де d - відстань між центрами сусідніх гребенів;
– поруч із закінченням обов'язково повинні проходити пара сусідніх гребенів;
– поруч з роздвоєнням обов'язково повинна проходити пара сусідніх западин.

2.6. Математична модель визначення точок, що утворені порізами та сторонніми предметами не є мінюціями і не впливають на порівняння.

Для виключення ненадійних точок визначимо ще одне положення локальних особливостей на вхідному растре.

На рисунку 2.11 показаний вид порізу або складки шкіри, при цьому виконується така умова:

$$(D_1 \cap L_1 \neq \emptyset) \wedge (D_2 \cap L_2 \neq \emptyset) \qquad (2.3)$$

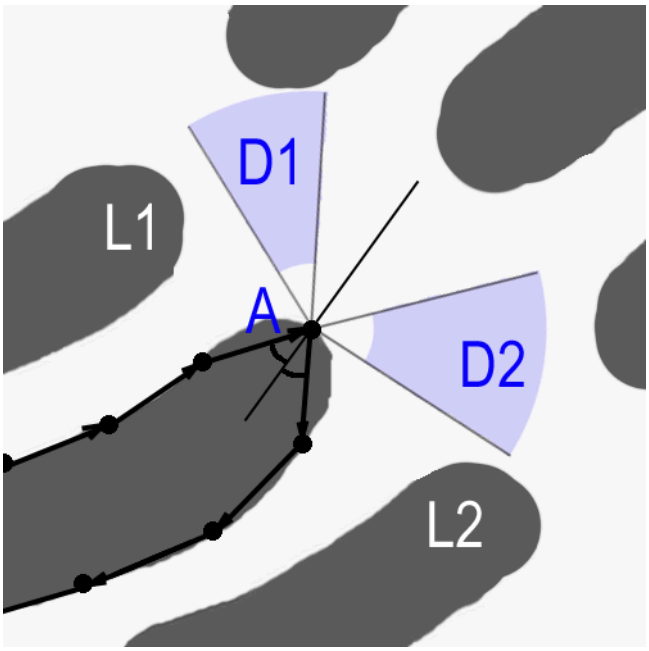

Рис. 2.13. «Поріз або складки шкіри»

D1, D2 – прилеглі області;

L1, L2 – ймовірні сусідні лінії папілярного візерунка.

Точки, утворені порізами та сторонніми предметами не є мінюціями і не впливають на порівняння.

Отриманий список сортується за умовою:

$$(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} < \Delta_1) \wedge (aMin < |a_j - a_i| < aMax) \vee (\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} < \Delta_2) \wedge (T_i = T_j) \quad (2.4)$$

де i, j – знайдені точки;

x, y – координати мінюції на растрі;

α – кут напрямку;

T – тип мінюції (роздвоєння або закінчення);

Δ1, Δ2, aMin, aMax – константи (встановлюються експериментально).

На рисунку 2.14 представлено алгоритм розв’язання задачі.

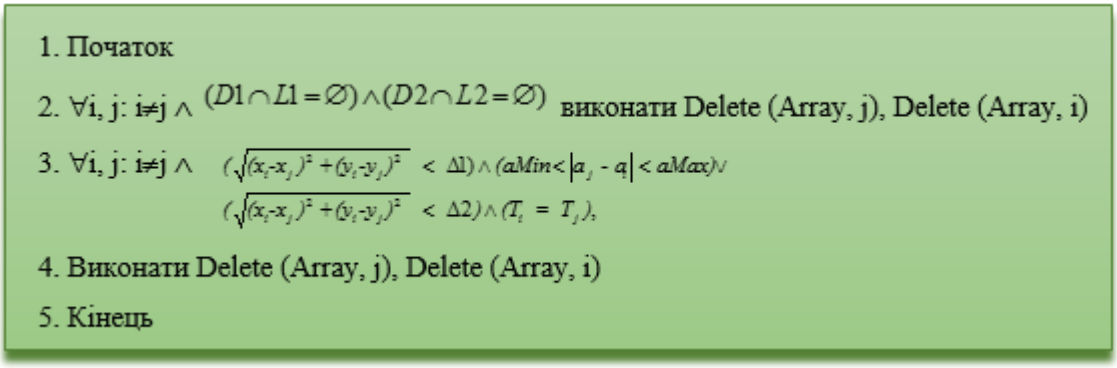

Рис. 2.14. Алгоритм задачі сортування списку абсолютних параметрів, виключення помилкових і ненадійних мінюцій.

РОЗДІЛ 3
РОЗРОБКА ПІДСИСТЕМИ АНАЛІЗУ ЗОБРАЖЕННЯ ПАПІЛЯРНОГО ВІЗЕРУНКУ

3.1. Функціональна схема підсистеми аналізу папілярного візерунка

Робота підсистеми реалізується наступними етапами:

– коригування вхідного образу, усунення дефектів і спотворень;

– пошук мінюцій і формування списку їх абсолютних параметрів;

– фільтрація отриманого списку параметрів;

Для вирішення поставлених завдань потрібні стандартні операції для роботи з масивом, які представлені у таблиці 2.2

Метою роботи є реалізація програми для виконання схеми, зображеної на рисунку 3.1.

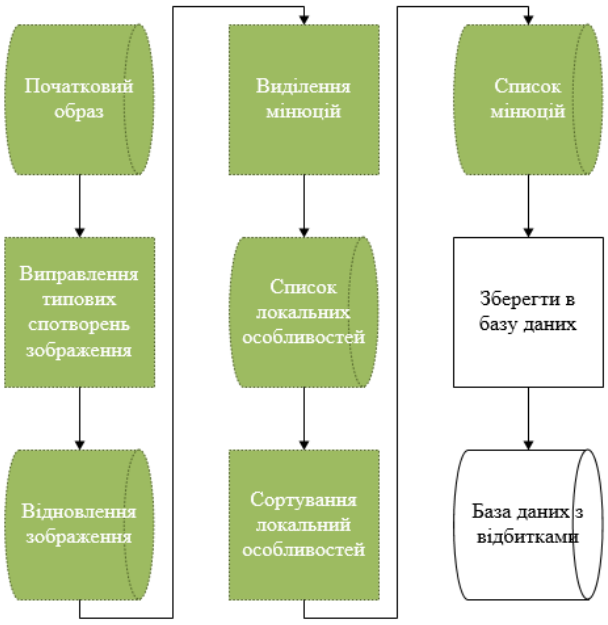

Рис. 3.1. Схема підсистеми аналізу

Вхідний інформацією є бітовий растр відбитка, отриманий за допомогою сканування дозволом 600dpi. Розширення бітового файлупо-замовчуванню *.bmp. Формат bmp (від слів BitMaP - бітовий масив) представляє з себе нестиснене (в основному), що дозволяє не вносити похибок, зображення. Формат bmp досить легко читається і виводиться в ОС Windows, в якій є спеціальні функції API.

Вхідний растр представлений форматом BMP, який має структуру представлену на рисунку 3.2 [11].

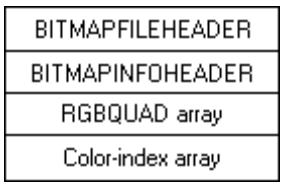

Рис. 3.2. Формат BMP

На початку повинен бути заголовок файлу – BITMAPFILEHEADER.

```
typedef struct tagBITMAPFILEHEADER
{
WORD bfType;
DWORD bfSize;
WORD bfReserved1;
WORD bfReserved2;
DWORD bfOffBits;
} BITMAPFILEHEADER, * PBITMAPFILEHEADER;
```

Далі йде структура – BITMAPINFOHEADER

```
typedef struct tagBITMAPINFOHEADER
{
DWORD biSize;
LONG biWidth;
LONG biHeight;
```

WORD biPlanes;

WORD biBitCount;

DWORD biCompression;

DWORD biSizeImage;

LONG biXPelsPerMeter;

LONG biYPelsPerMeter;

DWORD biClrUsed;

DWORD biClrImportant;

} BITMAPINFOHEADER, * PBITMAPINFOHEADER;

3.2. Програмна реалізація.

3.2.1. Підпрограма NextDotCW.

Підпрограма NextDotCW виробляє пошуку наступної точки на контурі лінії «за годинниковою стрілкою», завдяки їй організується обхід лінії по контуру. Призначена для реалізації алгоритму виправлення викривлень і використовується при пошуку мінюцій на вхідному образі. Схема підпрограми зображена на рисунку 3.1.

Синтаксис:

CPoint TFingPicture :: NextDotCW (const CPoint dot, int & vec)

Вхідні дані для даної підпрограми представлені:

CPoint dot – структура даних – точка {x, y} від якої потрібно знайти сусідню точку;

int vec – напрямок попереднього переходу при пошуку, vec ∈ [0..7], служить для прискорення пошуку:

COLORREF clMas [9] – масив квітів навколишніх точок.

Вихідні дані для даної підпрограми представлені:

CPoint incXY [8] – координати навколишніх точок;

CPoint newDot – знайдена точка, яка є суміжною з точкою dot. Перехід від точки dot до знайденої суміжній точці утворює обхід «за годинниковою стрілкою».

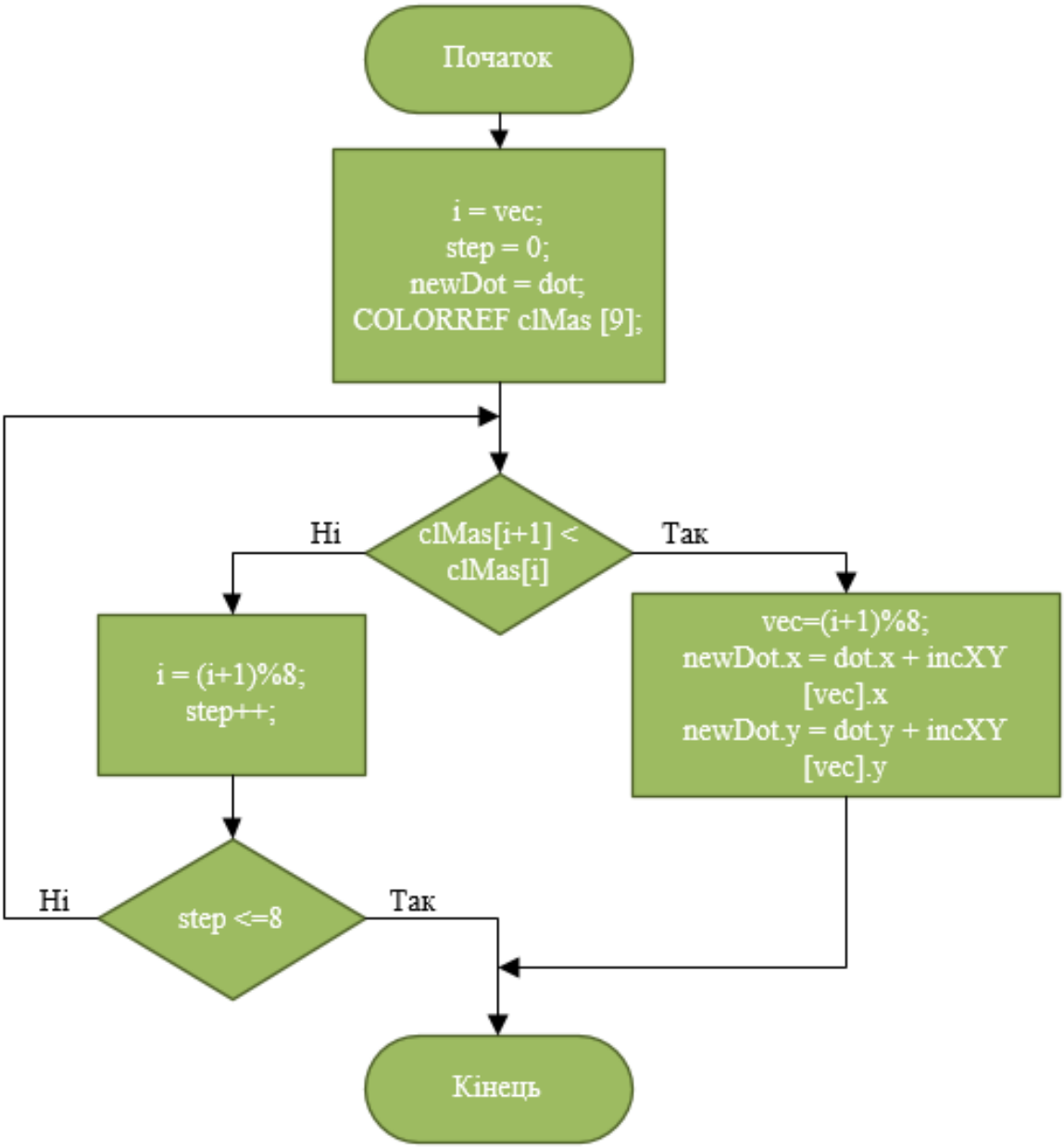

Рис. 3.3. Схема підпрограми «NextDotCW».

3.2.2. Підпрограма NextDotCCW

Підпрограма NextDotCCW виробляє пошуку наступної точки на контурі лінії «проти годинникової стрілки», завдяки їй організується обхід лінії по контуру. Призначена для реалізації алгоритму виправлення викривлень і використовується при пошуку мінюцій на вхідному образі. Схема підпрограми зображена на рисунку 3.4.

Синтаксис:

CPoint TFingPicture :: NextDotCCW (const CPoint dot, int & vec)

Вхідні дані для даної підпрограми представлені:

CPoint dot – структура даних - точка {x, y} від якої потрібно знайти сусідню точку;

int vec – напрямок попереднього переходу при пошуку, vec ∈ [0..7], служить для прискорення пошуку;

COLORREF clMas [9] – масив квітів навколишніх точок;

Вихідні дані для даної підпрограми представлені:

CPoint incXY [8] – координати навколишніх точок;

CPoint newDot – знайдена точка, яка є суміжною з точкою dot. Перехід від точки dot до знайденої суміжній точці утворює обхід «проти годинникової стрілки».

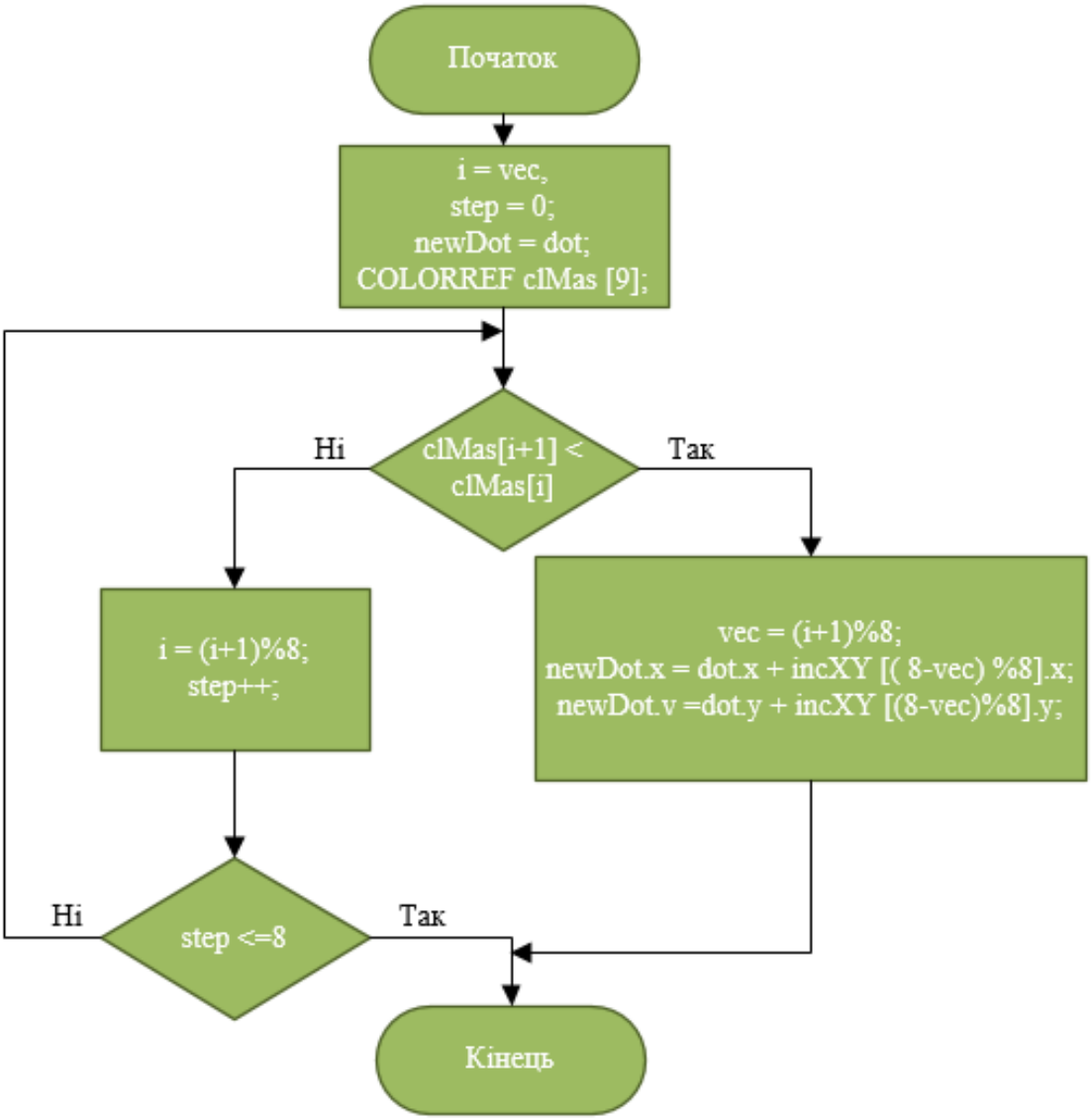

Рис. 3.4. Схема подпрограммы «NextDotCCW»

3.2.3. Підпрограма LockPick

Підпрограма Look Pic призначена для обробки завантаженого зображення і отримання з нього списку папілярних ліній. Кожна лінія визначається однією точкою {x, y}. Схема підпрограми зображена на рисунку 3.5.

Синтаксис:

list <TMapElDot> TAnalysePicture :: LookPic ()

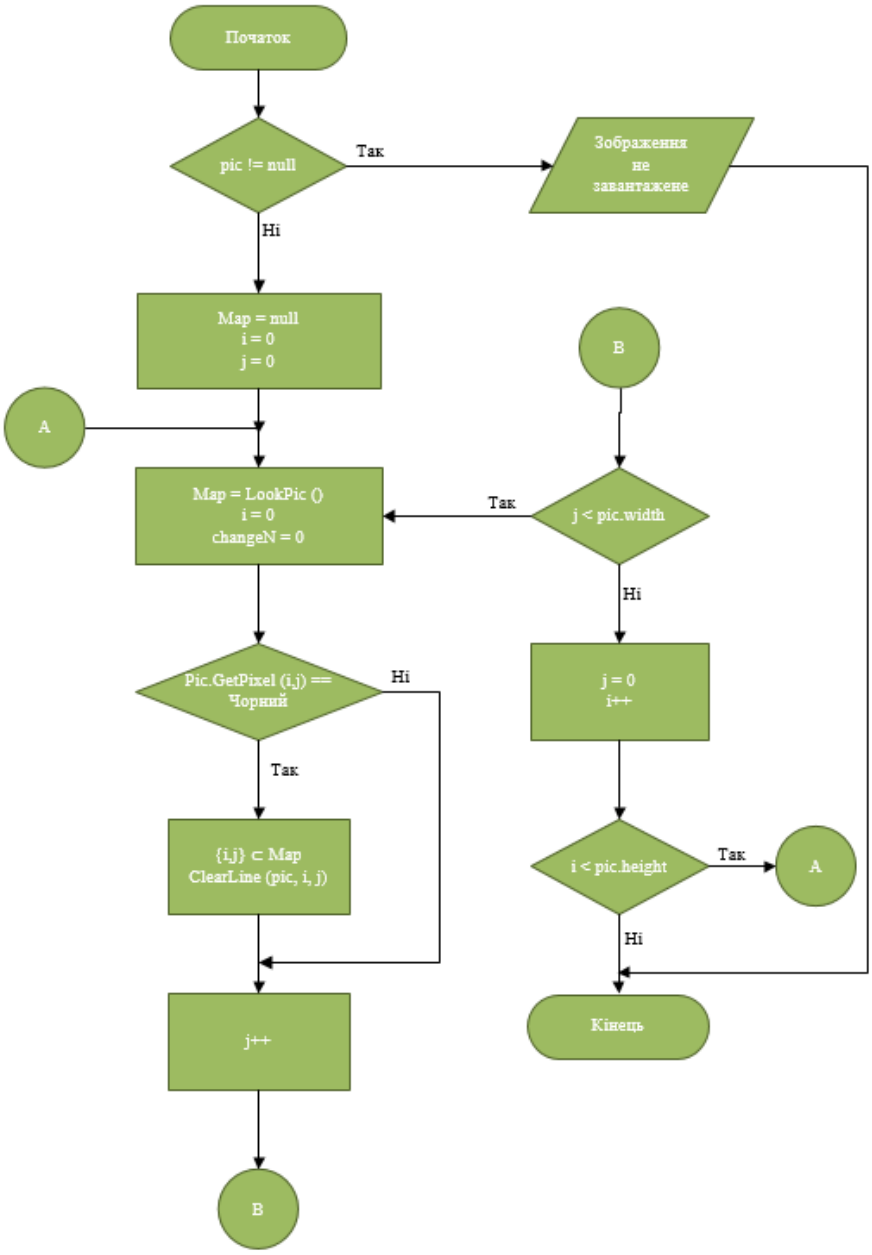

Рис. 3.5. Схема підпрограми «LookPic»

Вхідні дані для даної підпрограми представлені:

TFingPicture * pic – покажчик на бітовий образ в пам'яті, який був завантажений для обробки

Вихідні дані для даної підпрограми представлені:

list <TMapElDot> Map – список папілярних ліній на растрі.

Використовувані змінні:

Map – список оброблюваних ліній на папілярному візерунку, кожній лінії відповідає точка {x, y}.

Використовувані підпрограми:

Pic.GetPixel (x, y) – повертає колір пікселя з координатами {x, y} на растрі pic;

ClearLine (pic, x, y) – видалення області з кольором GetPixel (x, y) на растрі pic.

3.2.4. Підпрограма ChangeLine

Підпрограма ChangeLine призначена для модифікація лінії на растрі, проводить виправлення злипання і обривів. Схема підпрограми зображена на рисунку 3.6.

Синтаксис:

int TAnalysePicture :: ChangeLine (list <TMapElDot> :: iterator _dot, list <TMapElDot> & _map)

Вхідні дані для даної підпрограми представлені:

TFingPicture * pic – покажчик на бітовий образ в пам'яті, який був завантажений для обробки;

list <TMapElDot> :: iterator _dot – покажчик на поточну оброблювану лінію;

list <TMapElDot> & _map – список оброблюваних ліній на растрі.

Вихідні дані для даної підпрограми представлені:

int changeN – вироблену кількість виправлень на растрі;

TFingPicture * pic – в результаті обробки вхідної образ піддається змінам.

Використовувані змінні:

dot0, dot1 – точки належать контуру оброблюваної лінії. Початкове значення dot0 = _dot.

vec0, vec1 – локальні напрямки;

step – крок отримання подальшої точки;

alphaTest – зумовлена константа, яка визначає сильне викривлення контуру папілярної лінії.

Використовувані підпрограми:

GetVec (dot0, dot1) – напрям з точки dot0 в dot1;

NextDotCW (dot0, step) – отримання координат точки наступної через step точок.

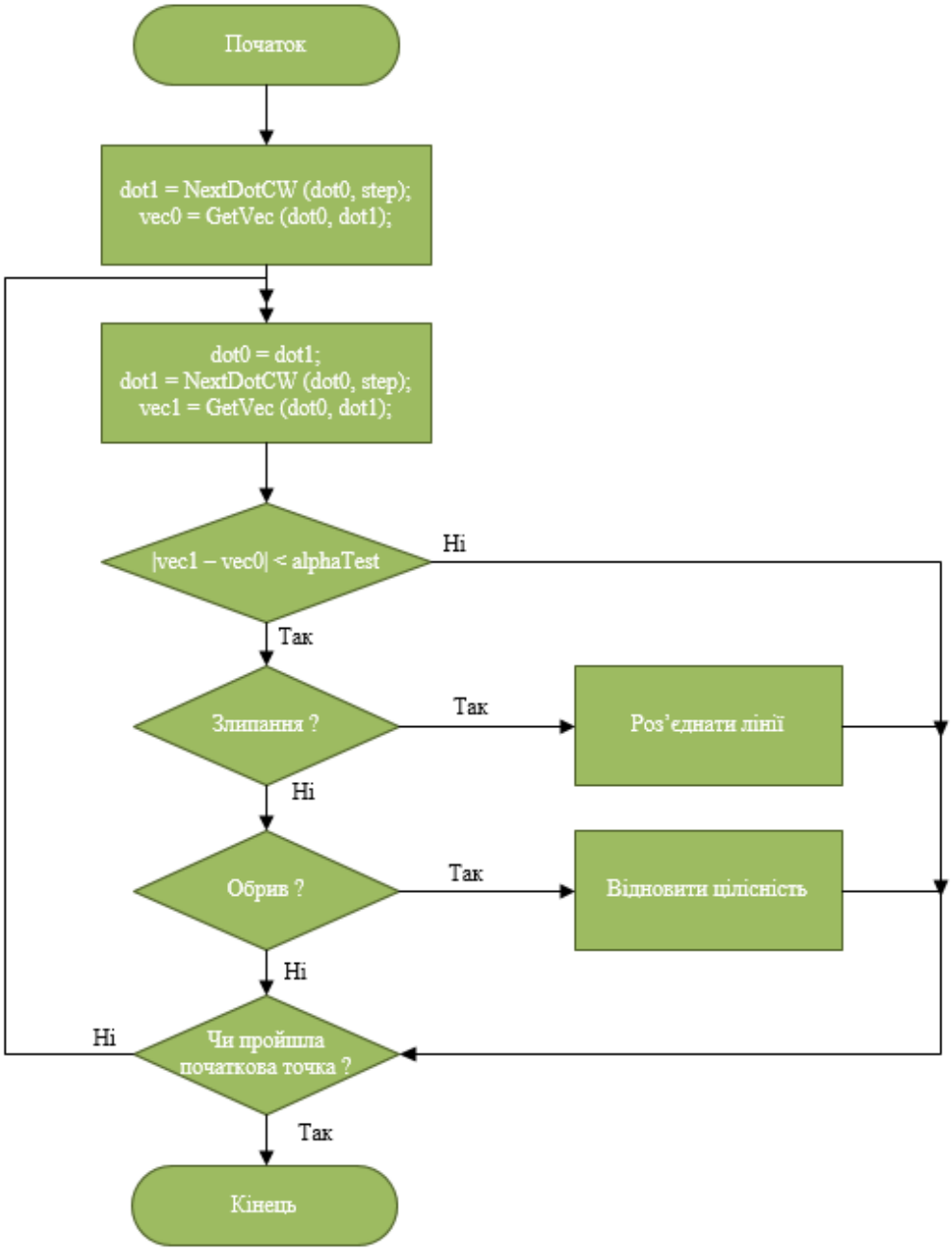

Рис. 3.6. Схема підпрограми «ChangeLine»

3.2.5. Підпрограма ReadFic

Підпрограма ReadPic призначена для пошуку локальних особливостей на растрі. Схема підпрограми зображена на рисунку. 3.7.

Синтаксис:

TAbsFing TAnalysePicture :: ReadPic (list <TMapElDot> :: iterator _dot)

Вхідні дані для даної підпрограми представлені:

TFingPicture * pic – покажчик на бітовий образ в пам'яті, який був завантажений для обробки;

list <TMapElDot> :: iterator _dot - покажчик на поточну оброблювану лінію.

Вихідні дані для даної підпрограми представлені:

TAbsFing absfing – список параметрів локальних особливостей.

Використовувані змінні:

Dot0, dot1 – точки належать контуру оброблюваної лінії. Початкове значення dot0 = _dot;

vec0, vec1 – локальні напрямки;

step – крок отримання подальшої точки;

alphaTest – зумовлена константа, яка визначає сильне викривлення контуру папілярної лінії.

Використовувані підпрограми:

GetVec (dot0, dot1) – напрям з точки dot0 в dot1;

NextDotCW (dot0, step) – отримання координат точки наступної через step точок.

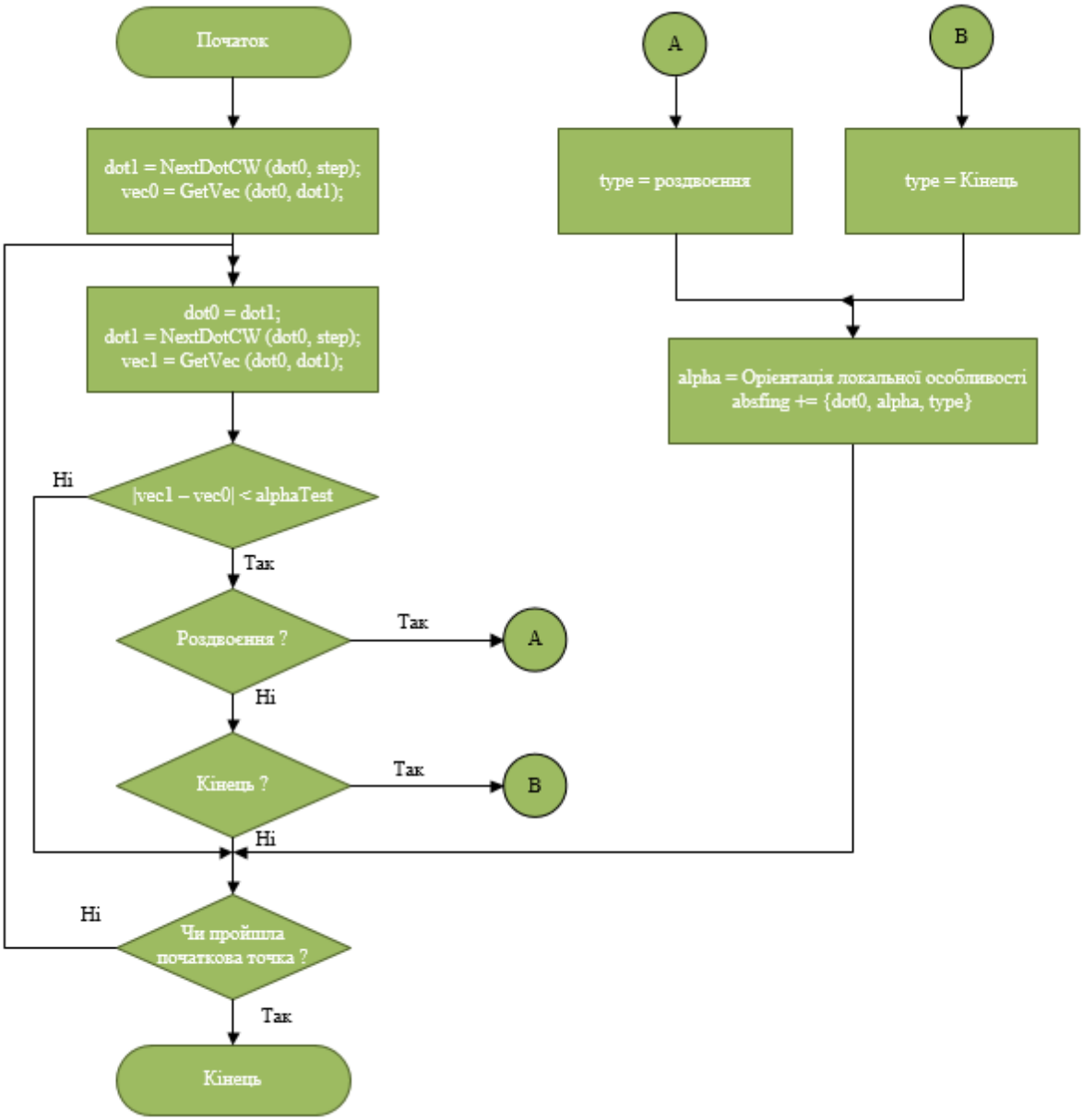

Рис. 3.7. Схема підпрограми «ReadPic»

3.2.6. Підпрограма Dots Filter

Підпрограма Dots Filter призначена для сортування списку знайдених локальних особливостей і виділення списку мінюцій. Схема підпрограми зображена на рисунку 3.8.

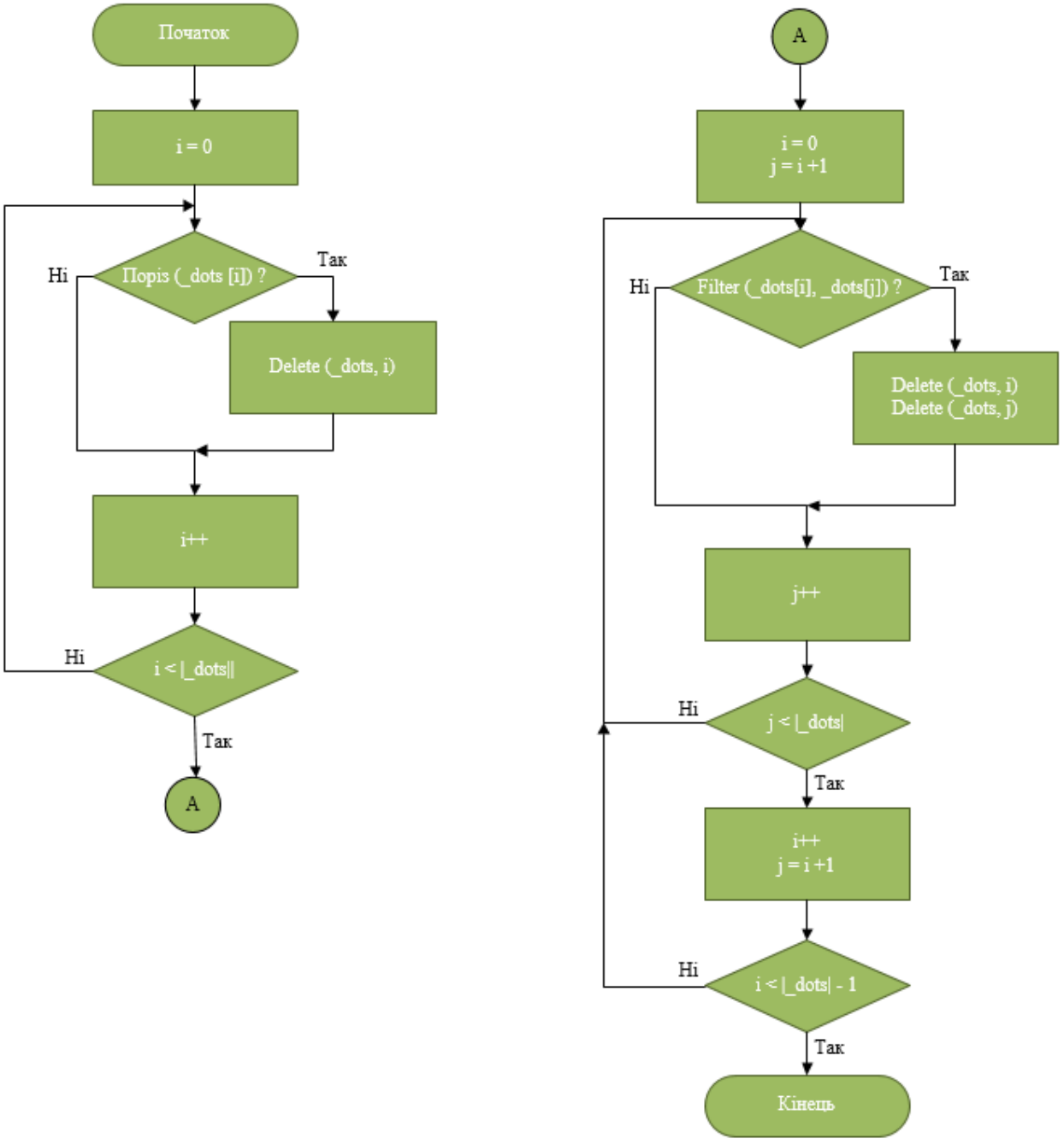

Рис. 3.8. Схема підпрограми «DotsFilter»

3.2.7. Підпрограма Analyse Picture

Підпрограма Analyse Picture призначена для обробки завантаженого зображення і отримання з нього об'єктного способу для подальшого зберігання та порівняння. Схема підпрограми зображена на рисунку. 3.9.

Синтаксис:

TAbsFing TAnalysePicture :: AnalysePicture ()

Вхідні дані для даної підпрограми представлені:

TFingPicture * pic – покажчик на бітовий образ в пам'яті, який був завантажений для обробки.

Дані для даної підпрограми представлені:

TAbsFing Ret – список координат мінюцій в абсолютних параметрах,

Використовувані змінні:

Map – список оброблюваних ліній на папілярному візерунку, кожній лінії відповідає точка {x, y};

changeN – зберігає кількість зроблених змін на растрі.

Використовувані підпрограми:

LookPic – повертає список ліній на відбитку;

ChangeLine (i, Map) – коригування лінії на растрі, позбавлення від злипання і обривів;

ReadPic – повертає список параметрів специфічних точок;

DotsFilter (Ret) – сортування специфічних точок.

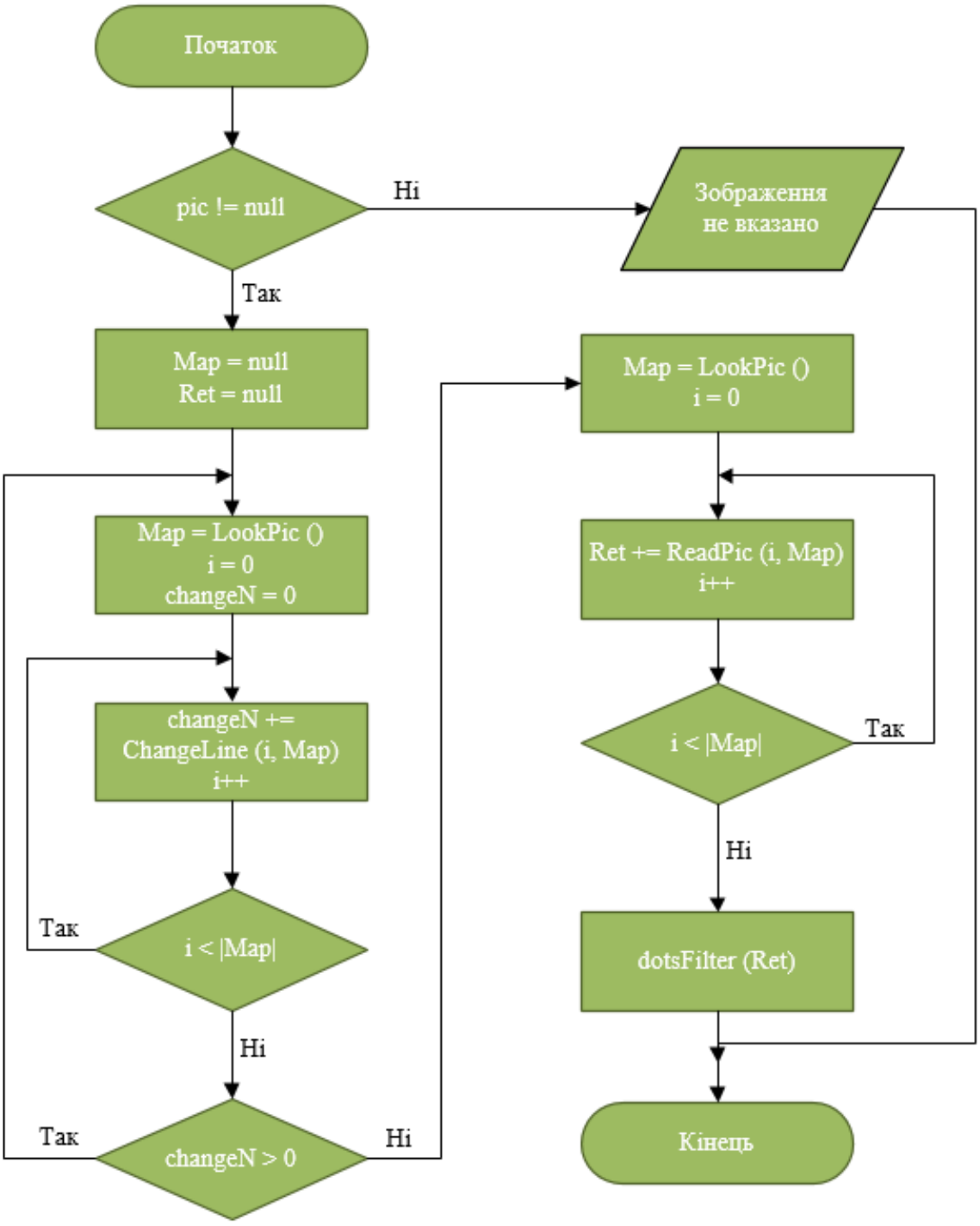

Рис. 3.9. Схема підпрограми «AnalysePicture»

3.3. Інтерфейс програмного додатку

Основною метою роботи програми є ідентифікація особи за відбитками пальців на основі порівняння структурного представлення папілярних візерунків. Програма розпізнавання особи за відбитками пальців має ідентифікатор FingerAnalyser і призначена для автоматичної ідентифікації особистості по папілярних узорів. Програма FingerAnalyser виконує наступні функції:

1) модифікація зображення, виправлення викривлень;

2) виділення локальних особливостей – мінюцій. Формування списку мінюцій в абсолютних параметрах;

3) сортування списку абсолютних параметрів, виключення помилкових і ненадійних мінюцій;

4) конвертація абсолютних параметрів у відносні, формування списку відносних параметрів.

Дана робота реалізує таке перетворення зображення, при якому дані про розташування унікальних особливостей зберігаються найбільш повно і з найменшим вмістом неправдивої інформації.

Створювана підсистема полегшить розробку алгоритмів обробки зображень, спростить аналіз експериментальних даних і виявлення загальних закономірностей.

Вікно програмного додатку має досить простий інтерфейс представлений на рисунку 3.10. Для завантаження вхідного зображення використовуємо кнопку «Open», для збереження результату – «Save», для закриття зображення – «Close».

Рис. 3.10. Інтерфейс програми FingerPrintAnalyser

Рис. 3.11. Приклади вхідних зображень

Програмне забезпечення за допомогою різних методів дозволяє покращити зображення відбитку. У даної підсистеми ми використовуємо такі методи: додавання контрастності (рисунок 3.12, згладжування гребенів (рисунок 3.13), згладжування (рисунок 3.14), бінаризація (рисунок 3.15), бінарне згладжування (рисунок 3.16) та видалення перетинів (рисунок 3.17). Все це потрібно для більш коректного аналізу зображення.

Рис. 3.12. Додавання контрастності

Рис. 3.13. Згладжування гребенів

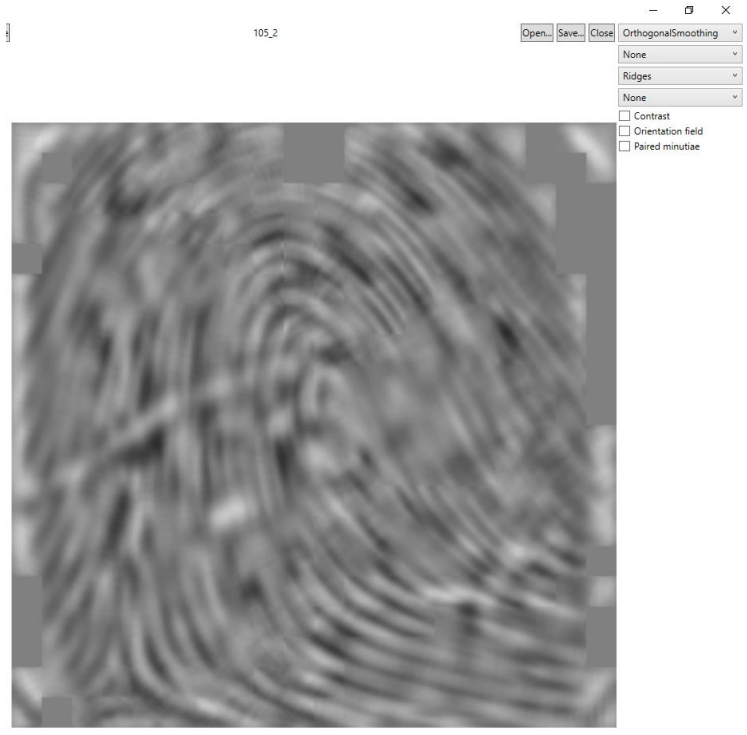

Рис. 3.14. Згладжування

Рис. 3.15. Бінаризація

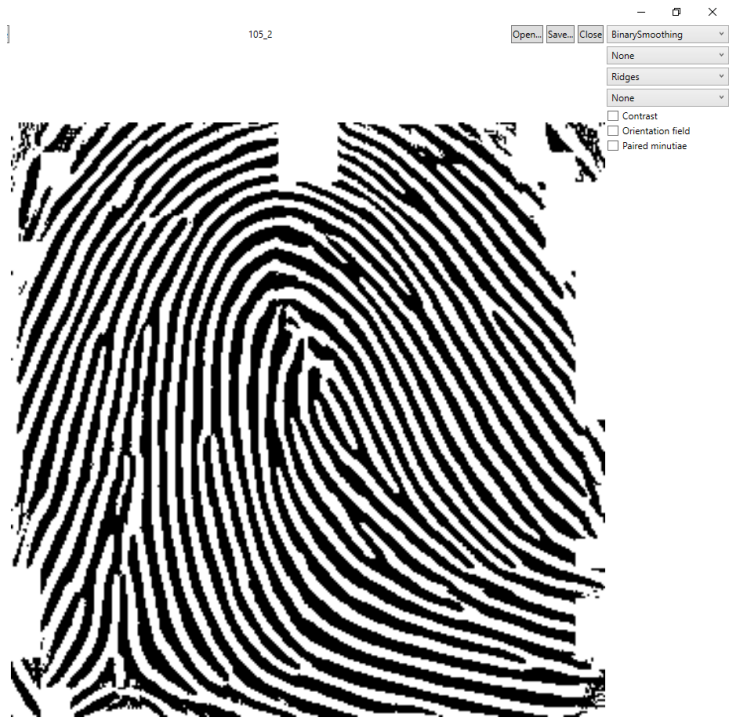

Рис. 3.16. Бінарне згладжування

Рис. 3.17. Видалення перетинів

Рис. 3.18. Унікальні особливості папілярного візерунка

Такий вигляд має програмний додаток для зрівняння двох відбитків (рисунок 3.19). Над цими відбитками проведемо ряд змін для покращення зображення: додання контрастності, бінаризація, бінарне згладжування та видалення перетинів (рисунок 3.20). Після чого можна приступати до пошуку унікальних особливостей папілярного візерунка (рисунок 3.21). Для пошуку схожих особливостей двох відбитків пальців використаємо функцію «Paired minutia», яка виділить знайдені однакові особливості папілярного візерунка (рисунок 3.22).

Рис. 3.19. Вхідні зображення

Рис. 3.20. Покращене зображення

Рис. 3.21. Знайдені особливості папілярного візерунка

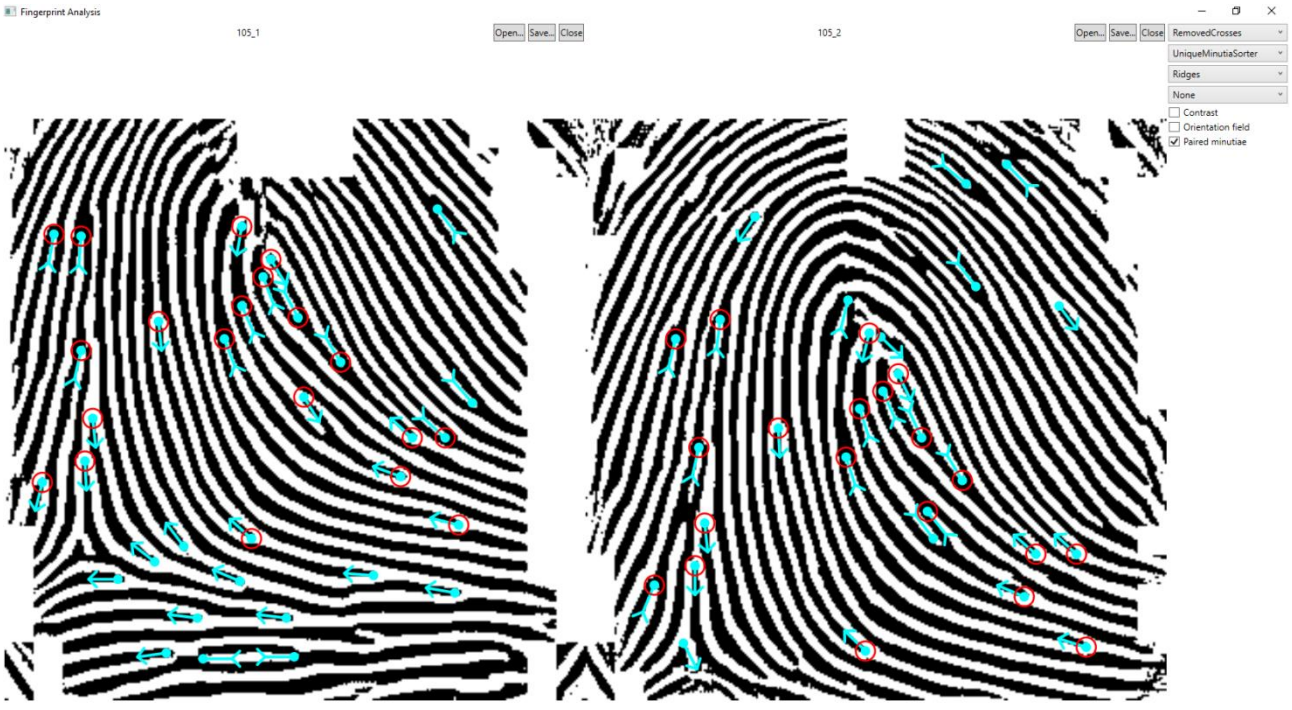

Рис. 3.22. Знайдені схожі мінюції

РОЗДІЛ 4
ЕКОНОМІЧНИЙ РОЗДІЛ

4.1. Розрахунок трудомісткості і вартості розробки програмного продукту

Вхідні дані:

- ✓ передбачуване число операторів – 2750
- ✓ коефіцієнт складності програми – 1,6
- ✓ коефіцієнт корекції програми в ході її розробки – 0,07
- ✓ часовая заработная плата программиста, грн/г – 300,0

У процесі створення ПЗ нормування праці ускладнено в силу творчого характеру праці програміста. Тому, трудомісткість розробки ПЗ розраховується на основі системи моделей з різною точністю оцінки.

$$t = t_u + t_a + t_n + t_{отл} + t_д, \text{ чол.-г} \qquad (4.1)$$

де t_u – затрати праці на дослідження алгоритму розв'язання задачі, чол.-г;

t_a – затрати праці на розробку блок-схеми алгоритму, чол.-г;

t_n – затрати праці на програмування по готовій блок–схемі, чол.-г;

$t_{отл}$ – затрати праці на налагодження програми на ЕОМ, чол.-г;

$t_д$ – затрати праці на підготовку документації по завданню, чол.-г.

Ці затрати праці визначаються через умовне число операторів при розробці ПЗ, в число яких входять ті оператори , які необхідно написати в процесі роботи над програмою з урахуванням можливих уточнень у постановці завдання і вдосконалення алгоритму .

Умовне число операторів в програмі обчислюється за формулою:

$$Q = qC(1 + p), \qquad (4.2)$$

де q - передбачуване число операторів q = 2750 ;

c - коефіцієнт складності програми c = 1,6;

p - коефіцієнт кореляції програми в ході її розробки p = 0,07.

Q = 2750 * 1,6 (1+ 0,07) = 4708

Затрати праці на вивчення опису завдання $t_и$ визначається з урахуванням уточнення опису та кваліфікації програміста.

$$t_и = \frac{QB}{(75..85)K} = \frac{4708*1,3}{77*1,2} = 66,24 \quad \text{чол.-год.,} \qquad (4.3)$$

де В - коефіцієнт збільшення затрат праці внаслідок недостатнього опису завдання:

$$B = 1,2. . . . 1,5;$$

К – коефіцієнт кваліфікації програміста, який визначається залежно від стажу роботи за даною спеціальністю. Він становить при стажі роботи, роки:

до 2 – 0,8;

від 2 до 3 – 1,0;

від 3 до 5 - 1,1 . . . 1,2;

від 5 до 7 - 1,3 . . . 1,4;

вище 7 – 1,5 . . . 1,6.

Затрати праці на розробку алгоритма для рішення задачі:

$$t_a = \frac{Q}{(20...25)K} = \frac{4708}{22*1,2} = 178,33 \quad \text{чол.-год.} \qquad (4.4)$$

Витрати на складання програми по готовій блок -схемі:

$$t_п = \frac{Q}{(20..25)K} = \frac{4708}{22*1,2} = 178,33 \quad \text{чол.-год.} \qquad (4.5)$$

Затрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4..5)K} = \frac{4708}{4*1,2} = 980,83 \text{ чол.-год.} \qquad (4.6)$$

$$t_{отл}^{K} = 1,5t_{отл} = 1,5*980,83 = 1471,24 \text{ чол.-год.} \qquad (5.7)$$

Витрати на підготовку документації:

$$t_д = t_{др} + t_{до} \quad \text{чол.-год.,} \qquad (4.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів:

$$t_{др} = \frac{Q}{(15...20)K} = \frac{4708}{17*1,2} = 230,78 \text{ чол.-год.} \quad (4.9)$$

$t_{до}$ - трудомісткість редагування, друку та оформлення документації:

$$t_{до} = 0,75 t_{др} = 0,75 * 230,78 = 173,08 \text{ чол.-год.} \quad (4.10)$$

$$t_{д} = t_{др} + t_{до} = 230,78 + 173,08 = 403,86 \text{ чол.-год.}$$

У підсумку отримуємо, що трудомісткість розробки ПЗ становить:

t = 66,24+178,33+178,33+980,83+403,86 = 1807,59 чол.-год.

4.2 Затрати на створення інформаційної системи.

Витрати на створення ПО (Кпо) включають витрати на заробітну плату виконавців програми (Зз/п), визначену множенням сумарної трудомісткості розробки ПО (t) на середню зарплату з нарахуваннями програміста і вартості машинного часу, необхідного для відладки програми на ЕОМ (Змв), визначеною виходячи з вартості 1-го машинного години конкретного типу ЕОМ, і витрат машинного часу на налагодження.

$$K_{ПО} = З_{ЗП} + З_{МВ} \text{ , грн.} \quad (4.11)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t * C_{ПР} = 1807,59 * 300,0 = 542277 \text{ грн.,} \quad (4.12)$$

де t - загальна трудомісткість, чол.-г.;

$C_{пр}$ - середня годинна заробітна плата програміста, грн./год;

$C_{пр}$ =300,0 грн./год.

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{мв} = t_{отл} * C_{мч} = 980,83 * 1,2 = 1176,99 \text{ грн.,} \quad (4.13)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год.;

$C_{мч}$ – вартість машинного часу ЕОМ, грн./год.

$$Кпо = 542277 + 1176,99 = 543453,99 \text{ грн.} \quad (4.14)$$

Визначені таким чином витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУТП . Очікуваний період розробки ПЗ:

$$T = \frac{t}{B_k \bullet F_p} \text{ міс.,} \quad (4.15)$$

де B_k - кількість виконавців;

F_p - місячний фонд робочого часу (при 40-ка годинному робочому тиждні F_p=176 годин).

$$T = \frac{1807,59}{1 * 176} \approx 10 \text{ мес.}$$

Таким чином, період розробки програми складе приблизно 10 місяців.

4.3 Маркетингові дослідження ринку

Система ідентифікації особи за відбитками пальців реалізує визначення особистості на основі біометричних параметрів людського тіла, а саме побудова відбитків пальців. Система призначена для обробки графічних зображень відбитків. Система дозволяє порівняти кілька відбитків один з одним по виділеним локальним особливостям. Локальними особливостями є мінюції і їх відносні параметри (розташування одних мінюцій щодо всіх інших), що гарантує незалежність порівняння від паралельного перенесення і обертання.

Програмний продукт знайде застосування в різних прикладних системах, включаючи:

1) системи цивільного ідентифікації;

2) криміналістичні системи ідентифікації;

3) великомасштабні комерційні додатки.

Системи цивільного ідентифікації включають в себе:

– водійські посвідчення;
– національні ідентифікаційні карти громадян;
– реєстрація виборців;
– реєстрація для соціальних програм;
– імміграційна реєстрація, візи;
– ідентифікація співробітників державних установ.

Криміналістичні системи ідентифікації включають:

– чи знаходиться даний громадянин в розшуку?;
– колишні судимості;
– реєстрація укладених / контроль доступу;
– мобільні та віддалені програми;
– обробка слідів відбитків пальців, отриманих з місць злочину.

Великомасштабні комерційні додатки включають:

– доступ до web-ресурсів, електронна комерція;
– доступ для користувачів і співробітників;
– фінансові послуги, перевірка оплати;
– доступ в будівлі і приміщення;
– програми лояльності.

ВИСНОВКИ

На сьогоднішній день існують готові системи для ідентифікації особи, що володіють високим ступенем захисту, швидкодією, а також зручністю в застосуванні. Однак жодна з існуючих розробок не дає об'єктного опису і методу порівняння відбитків. Всі розробки є унікальними, мають власні нововведеннями, «ноу-хау» і складають комерційну таємницю.

Серед програмних продуктів, присвячених ідентифікації по папілярних узорів, можна виділити основні можливості:

1) програми реалізують можливість доступу з використанням відбитка;

2) можлива обробка стандартними функціями (яскравість, контрастність, зміна розміру);

3) розпізнавання символів;

4) жодна програма не дозволяє скорегувати зображення, ґрунтуючись на типові характеристики відбитка, дати об'єктне опис відбитка, а також дати можливість застосувати алгоритми обробки окремо для власних завдань.

У зв'язку з зазначеними особливостями існуючих програмних засобів і в силу того, що застосування біометричних способів дозволяє збільшити захищеність і зручність користування системами для більшості розробників буде зручним використання готового модуля роботи з відбитками пальців. Тому актуальною є розробка системи, що володіє відкритим кодом і дозволяє проводити структурний опис папілярного візерунка. Можливість отримувати його об'єктне опис і порівняння.

Це завдання вирішує система розпізнавання особи за папілярним візерунком.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. JAIN A. K. AND PRABHKAR S. 2001. Fingerprint Matching Using Minutiae and Texture Features. Proceeding of International Conference on Image Processing (ICIP), pp. 282-285.

2. AGGARWAL G., RATHA N. K., TSAI-YANG J., AND BOLLE R. M. 2008. Gradient based textural characterization of fingerprints. In proceedings of IEEE International conference on Biometrics: Theory, Applications and Systems.

3. CHIKKERUR S., PANKANTI S., JEA A., AND BOLLE R. 2006. Fingerprint Representation using Localized Texture Features. The 18th International Conference on Pattern Recognition

4. JHAT Z. A., MIR A. H. AND RUBAB S. 2011. Fingerprint Texture Feature for Discrimination and Personal Verification. International Journal of Security and its Applications, Vol. 5, No. 3

5. V. Dixit, Deepti Singh, Parul Raj, M. Swathi, P. Gupta, kd-tree based fingerprint identification system 01/2008; DOI:10.1109/IWASID.2008.4688340

6. ZHENGU O., FENG J., SU F., AND CAI A., 2006. Fingerprint Matching with Rotation-Descriptor Texture Features. The 18th International Conference on Pattern Recognition, pp. 417-420.

7. Johan De Boer, Asker M Bazen, Sabih H Gerez, Indexing fingerprint databases based on multiple features Journal of The Acoustical Society of America- J ACOUST SOC AMER. 12/2001;

8. J.H. Wegstein, A Semi-automated Single Fingerprint Identification System. NBS Technical Note 481, April 1969.

9. J.H. Wegstein, The M40 Fingerprint Matcher. NBS Technical Note 878, July 1975.

10. R.T. Moore, Results of Fingerprint Image Quality Experiments. NBS Technical Report NBSIR 81-2298, June 1981.

11. R.T. Moore Automated Fingerprint Identification Systems - Benchmark

Test of Relative Performance. American National Standard ANSI/IAI 1-1988, February 1988.

12. C. Wilson, M. Garris, C. Watson, A, Hicklin, Studies of Fingerprint Matching Using the NIST Verification Test Bed (VTB). Technical Report NISTIR 7020, July 2003.

13. C. Watson, C. Wilson, M. Indovina, R. Snelick, K. Marshall, Studies of One-toOne Matching with Vendor SDK Matchers. Technical Report NISTIR 7119, July 2004.

Додаток А

Лістинг програми

```
bic char(11) not null
unique,
stat char(1)
default 'C'
check (stat in ('C','D','5'))
)
go
grant select on bic to publicgodrop table curcgocreate table curc
(
curc_id integer not null primary key
DEFAULT autoincrement,
curc char(3) not null,
summa char(17) default '20,00' not null,
Comments varchar(50) null,
bic_id integer null references bic(bic_id))gogrant select on curc to
publicgo
drop table mt_103_rcvdgocreate table mt_103_rcvd(
id_103 integer not null primary key
DEFAULT autoincrement,
f_20 char(16) not null,
vald char(8) not null,
curc char(3) not null,
amnt char(17) not null,sndr char(12) not null,
text varchar(32767) not null,
ms_dbky char(18) not null
)go
grant select on mt_103_rcvd to publicgodrop table mt_910gocreate table
mt_910(
id_910 integer not null primary key
DEFAULT autoincrement,f_20 char(16) not null,
f_21 char(16) not null
default 'NON REFF',
vald char(8) not null,
curc char(3) not null,
amnt char(17) not null,sndr char(12) not null,
f_72 varchar(210) null,
ms_dbky char(18) not null)gogrant select on mt_910 to publicgo
drop table mt_202go
```

```
create table mt_202(
id_202 integer not null primary key
DEFAULT autoincrement,f_20 char(16) not null,
f_21 char(16) not null
default 'NON REFF',
vald char(8) not null,
curc char(3) not null,
amnt char(17) not null,sndr char(12) not null,
f_72 varchar(210) null,
ms_dbky char(18) not null)gogrant select on mt_202 to publicgo
drop table mt_950go
create table mt_950(
id_950 integer not null primary key
DEFAULT autoincrement,f_20 char(16) not null,
f_21 char(16) not null
default 'NON REFF',
f_61 varchar(98) null,
ms_dbky char(18) not null)gogrant select on mt_950 to publicgo
drop table mt_n9ngo
create table mt_n9n(
n9n_id integer not null primary key
DEFAULT autoincrement,f_20 char(16) not null,
f_21 char(16) not null
default 'NON REFF',
text varchar(32767) not null,
ms_dbky char(18) not null)gogrant select on mt_n9n to publicgo
drop table mt_191_sendgocreate table mt_191_send(
id_191 integer not null primary key
DEFAULT autoincrement,
f_20 char(16) not null,
f_21 char(16) not null,
vald char(8) not null,
curc char(3) not null,
amnt char(17) not null,sndr char(12) not null,
stat char(1) default 'N' not null
check (stat in ('N','P')),
ms_dbky char(18) not null,
f_72 varchar(210) null,
id_103 integer references mt_103_rcvd(id_103) null,
id_910 integer references mt_910(id_910) null,
id_202 integer references mt_202(id_202) null,
```

```
id_950 integer references mt_950(id_950) null,
n9n_id integer references mt_n9n(n9n_id) null
)
gogrant select on mt_191_send to public
go
drop table dep
go
create table dep
(
id_dep integer not null primary key
default autoincrement,
DepName varchar(30) null unique,
Comments varchar(225) null,
)
go
drop table empl
go
create table empl
(
      id_empl integer not null primary key
      default autoincrement,
      FName varchar(20) null,
      LName varchar(20) null,
      id_dep integer references dep(id_dep)
)
go
drop table ugroup
go
create table ugroup
(
      id_group integer not null primary key
      default autoincrement,
      GName varchar(30) not null unique
      default ('user')
      check (GName in('user','admin','guest')),
      Comments varchar(255) null
)
go
drop table rights
go
create table rights
```

```
(
 id_right integer not null primary key
 default autoincrement,
 flags char(5) not null
 default 'R'
 check (flags in ('R','RW','IUD','RWUID')),
 id_group integer references ugroup(id_group),
 id_secr integer references secrets(id_secr)
)
go
create table secrets
(
      id_secr integer not null primary key
      default autoincrement,
      uid varchar(10) not null unique
      default ('chrg_use'),
      pwd varchar(10)not null
      default ('chrg_use'),
      datein char(8) null,
      id_empl integer references empl(id_empl)
)
go
create unique index bicid on bic(bic)gocreate index camnt_103 on
mt_103_rcvd (curc,amnt)go
create index f_20_103 on mt_103_rcvd (f_20)go
create index f_20_191 on mt_191_send (f_20)gocreate index f_20_910 on
mt_910 (f_20)go
create index f_20_202 on mt_202 (f_20)go
create index f_20_950 on mt_950 (f_20)go
create index f_20_n9n on mt_n9n (f_20)go

/* SQL Anywhere 3/1/96*/set option date_format='Mmm dd yyyy hh:mmaa'goset
option date_order = 'MDY'goset option scale = 2go

drop procedure add_mt_103_rcvd
go

create procedure add_mt_103_rcvd(
in inf_20 char(16),
in invald char(8),
in incurc char(3),
```

```
in inamnt char(17),
in insndr char(12),
in intext varchar(32767),
in inms_dbky char(18))
begin
insert into mt_103_rcvd(f_20,vald,curc,amnt,sndr,text,ms_dbky)
values(inf_20,invald,incurc,inamnt,insndr,intext,inms_dbky)
end
go

drop procedure add_mt_910
go

create procedure add_mt_910
(
in inf_20 char(16),
in inf_21 char(16),
in invald char(8),
in incurc char(3),
in inamnt char(17),
in insndr char(12),
in inf_72 varchar(210),
in inms_dbky char(18)
)
begin
insert into mt_910(f_20,f_21,vald,curc,amnt,sndr,f_72,ms_dbky)
values(inf_20,inf_21,invald,incurc,inamnt,insndr,inf_72,inms_dbky)
end
go
drop procedure add_mt_202
go
create procedure add_mt_202(
in inf_20 char(16),
in inf_21 char(16),
in invald char(8),
in incurc char(3),
in inamnt char(17),
in insndr char(12),
in inf_72 varchar(210),
in inms_dbky char(18))
begin
```

```
insert into mt_202(f_20,f_21,vald,curc,amnt,sndr,f_72,ms_dbky)
values(inf_20,inf_21,invald,incurc,inamnt,insndr,inf_72,inms_dbky)
end
go
drop procedure add_mt_950
go
create procedure add_mt_950(
in inf_20 char(16),
in inf_21 char(16),
in inf_61 varchar(98),
in inms_dbky char(18))
begin
insert into mt_950(f_20,f_21,f_61,ms_dbky)
values(inf_20,inf_21,inf_61,inms_dbky)
end
go
drop procedure add_mt_n9n
go
create procedure add_mt_n9n(
in inf_20 char(16),
in inf_21 char(16),
in intext varchar(32767),
in inms_dbky char(18)
)
begin
insert into mt_n9n(f_20,f_21,text,ms_dbky)
values(inf_20,inf_21,intext,inms_dbky)
end
go
drop procedure add_mt_191_send
go
create procedure add_mt_191_send(
in inf_20 char(16),
in inf_21 char(16),
in invald char(8),
in incurc char(3),
in inamnt char(17),
in insndr char(12),
in instat char(1),
in inms_dbky char(18),
in inf_72 varchar(210)
```

```
)
begin
insert into mt_191_send(f_20,f_21,vald,curc,amnt,sndr,stat,ms_dbky,f_72)
values(inf_20,inf_21,invald,incurc,inamnt,insndr,instat,inms_dbky,inf_72)
end
go
drop procedure add_dep
go
create procedure add_dep(
in inDepName varchar(30),
in inComments varchar(225)
)
begin
      insert into dep(DepName,Comments)
      values(inDepName,inComments)
end
go
drop procedure upd_dep
go
create procedure upd_dep(
in oldDepName varchar(30),
in inDepName varchar(30),
in inComments varchar(225)
)
begin
      declare in_id integer;
      select id_dep into in_id
      from dep
      where  DepName = oldDepName;
      update dep
      set DepName=inDepName,Comments=inComments
      where id_dep = in_id
end

go
drop procedure del_dep

go
create procedure del_dep(
in inDepName varchar(30)
)
```

```
begin
declare in_id integer;
select id_dep into in_id
from dep
      where DepName=inDepName;
delete from dep
where id_dep = in_id
end
go
drop procedure add_empl
go
create procedure add_empl(
in inFName varchar(20),
in inLName varchar(20),
in inDepName varchar(30)
)
begin
      declare in_id integer;
      select id_dep into in_id
      from dep
      where DepName=inDepName;
      insert into empl(FName,LName,id_dep)
      values(inFName,inLName,in_id)
end
go
drop procedure upd_empl
go
create procedure upd_empl(
in inid_empl integer,
in inFName varchar(20),
in inLName varchar(20),
in inDepName varchar(30))
begin
      declare inid integer;
      select id_dep into inid
      from dep
      where DepName=inDepName;
      update empl
      set FName=inFName,LName=inLName,id_dep=inid
      where id_empl = inid_empl;
end
```

```
go
drop procedure del_empl
go
create procedure del_empl(
in inid_empl integer
)
begin
      delete from empl
      where id_empl = inid_empl
end
go
drop procedure add_ugroup
go
create procedure add_ugroup(
in inGName varchar(30),
in inComments varchar(255)
)
begin
      insert into ugroup(GName,Comments)
      values(inGName,inComments)
end
go
drop procedure upd_ugroup
go
create procedure upd_ugroup(
in oldGName varchar(30),
in inid_group integer,
in inGName varchar(30),
in inComments varchar(255)
)
begin
      declare in_id integer;
      select id_group into in_id from ugroup
      where GName=oldGName;
      update ugroup
      set GName = inGName,Comments = inComments
      where id_group=in_id
end
go
drop procedure del_ugroup
go
```

```
create procedure del_ugroup(
in inid_group integer
)
begin
      delete from ugroup
      where id_group = inid_group
      end
go
drop procedure add_rights
go
create procedure add_rights(
in inflags char(5),
in inGName varchar(30),
in inUid varchar(10)
)
begin
      declare in_id integer;
      declare inid_secr integer;
      select id_group into in_id from ugroup
      where GName=inGName;
      select id_secr into inid_secr from secrets
      where uid=inUid;
      insert into rights(flags,id_group,id_secr)
      values(inflags,in_id,inid_secr)
end
go
drop procedure upd_rights
go
create procedure upd_rights(
in inflags char(5),
in inGName varchar(30)
)
begin
      declare in_id integer;
      declare in_id_r integer;
      select id_group into in_id
      from ugroup
      where GName = inGName;
      select id_right into in_id_r
      from rights
      where id_group = in_id;
```

```
      update rights set flags=inflags
      where id_right = in_id_r
end
go

drop procedure del_rights
go
create procedure del_rights(
in inid_right integer
)
begin
      delete from rights
      where id_right=inid_right
end
go
drop procedure add_secrets
go
create procedure add_secrets(
in inuid varchar(10),
in inpwd varchar(10),
in indatein char(8),
in inid_empl integer
)
begin
      insert into secrets(uid,pwd,datein,id_empl)
      values(inuid,inpwd,indatein,inid_empl)
end
go
drop procedure upd_secrets
go
create procedure upd_secrets(
in inGName varchar(30),
in inid_secr integer
)
begin
      declare inid_right integer;
      declare in_id integer;
      select id_group into in_id
      from ugroup
      where GName = inGName;
      select id_right into inid_right
```

```
      from rights
      where id_group = in_id;
    update secrets set
    id_right=inid_right
    where id_secr=inid_secr
end
go
drop procedure del_secrets
go
create procedure del_secrets(
in inid_secr integer
)
begin
      delete from secrets
      where id_secr=inid_secr
end
go
drop procedure add_bic
go
create procedure add_bic
(
inbic char(11),
instat char(1)
)
begin
      insert into bic(bic,stat)
      values(inbic,instat)
end
go

drop procedure del_bic
go
create procedure del_bic
(
inbic char(11)
)
begin
      delete from bic
      where bic = inbic;
end
go
```

```
drop procedure upd_bic
go
create procedure upd_bic
(
inbic_id integer,
inbic char(11),
instat char(1)
)
begin
      update bic set bic = inbic,stat=instat
      where bic_id=inbic_id;
end
go
drop procedure add_curc
go
create procedure add_curc
(
incurc char(3),
insumma char(17),
inComments varchar(50),
inbic char(11)
)
begin
      declare in_id integer;
      select bic_id into in_id
      from bic
      where bic = inbic;
      insert into curc(curc,summa,Comments,bic_id)
      values(incurc,insumma,inComments,in_id)
end
go
drop procedure upd_curc
go
create procedure upd_curc
(
incurc_id integer,
incurc char(3),
insumma char(17),
inComments varchar(50),
inbic char(11)
)
```

```
begin
      declare in_id integer;
      select bic_id into in_id from bic
      where bic = inbic;
      update curc
      set curc=incurc,summa=insumma,Comments=inComments,bic_id=in_id
      where curc_id=incurc_id
end
go
drop procedure del_curc
go
create procedure del_curc
(
incurc_id integer
)
begin
      delete from curc
      where curc_id = incurc_id;
end
go
create trigger before_del_bic
before delete on bic
referencing old as del_bic_id
for each row
begin
      delete from curc
      where curc.bic_id=del_bic_id;
end
go
create trigger after_update_bic
after update of bic_id on bic
referencing old as old_bic_id
new as new_bic_id
for each row
begin
      update curc
      set curc.bic_id=new_bic_id
      where curc.bic_id=old_bic_id
end
go
create trigger before_del_secrets
```

```
before delete on secrets
referencing old as del_empl_id
for each row
begin
      delete from empl
      where empl.id_empl=del_empl_id
end
go
create trigger before_del_empl
before delete on empl
referencing old as del_dep_id
for each row
begin
      delete from dep
      where dep.id_dep=del_dep_id
end
go

create procedure read_103(in skey varchar(18))
result ("ms_trnf" char(16),"ms_vald" char(8),"ms_curc" char(3),"ms_amnt"
char(17),"ms_sndr" char(12),"ms_dbky" char(18), "ms_text" char(500))
begin
      select distinct
ms_trnf,ms_vald,ms_curc,ms_amnt,ms_sndr,ms_dbky,ms_text from ms_
      where ms_stat = 'R'
      and ms_dirc = 'O'
      and ms_mtyp = '103'
      and ms_dbky like skey;
end

create procedure read_103_all(in skey varchar(18))
result ("ms_text" char(574))
begin
      select distinct ms_trnf+"->"+ms_vald+"->"+ms_curc+"->"+ms_amnt+"-
>"+ms_sndr+"->"+ms_dbky+"->"+ms_text from ms_
      where ms_stat = 'R'
      and ms_dirc = 'O'
      and ms_mtyp = '103'
      and ms_dbky like skey;
end
```

```
/*по db_key возвращает всё сообщение, без его головы (первые 500 байт)*/
create procedure rmte(in skey varchar(18),out rslt_text varchar(32767))
begin
    declare err_notf
    EXCEPTION FOR SQLSTATE '02000';
    declare tmp_text varchar(32767);
    declare cselmx cursor for
     select mx_text from mx_
     where mx_dbky=skey
     order by mx_sgmt asc;
    set rslt_text = '';
    open cselmx;
    mxloop:
    loop
        fetch next cselmx
            into tmp_text;
        if sqlstate = err_notf then
            leave mxloop;
        end if;
        set rslt_text=rslt_text+tmp_text;
    end loop mxloop;
close cselmx;
end

create procedure rmte_sel(in skey varchar(20))
begin
    declare rslt_text varchar(32767);
    declare err_notf
    EXCEPTION FOR SQLSTATE '02000';
    declare tmp_text char(500);
    declare cselmx cursor for
     select mx_text from mx_
     where mx_dbky=skey
     order by mx_sgmt asc;
    set rslt_text = '';
    open cselmx;
    mxloop:
    loop
        fetch next cselmx
            into tmp_text;
        if sqlstate = err_notf then
```

```
      begin
          set rslt_text=rslt_text + '';
             leave mxloop;
      end
         end if;
         set rslt_text=rslt_text+tmp_text;
     end loop mxloop;
close cselmx;
select rslt_text;
end
create procedure read_103_all(in skey varchar(20))
result (fnd_dbky varchar(20), rslt_text char(600),end_text
varchar(32767))
begin
      select distinct ms_dbky,ms_text,(select * from rmte_sel(ms_dbky))as
end_text from ms_
      where ms_stat = 'R'
      and ms_dirc = 'O'
      and ms_mtyp = '103'
      and ms_dbky like skey;
end

create procedure read_103_all(in skey varchar(20))
result (fnd_dbky varchar(20), all_text varchar(32767))
begin
      select distinct ms_dbky,ms_text+(select * from rmte_sel(ms_dbky))
from ms_
      where ms_stat = 'R'
      and ms_dirc = 'O'
      and ms_mtyp = '103'
      and ms_dbky like skey;
end

create procedure read_103_all(in skey varchar(20))
result (fnd_dbky varchar(20), all_text varchar(32767))
begin
     declare rslt_text varchar(32767);
      select distinct ms_dbky,ms_text+(select rslt_text from
rmte_sel(ms_dbky)) from ms_
      where ms_stat = 'R'
      and ms_dirc = 'O'
```

```
      and ms_mtyp = '103'
      and ms_dbky like skey;
end

SELECT SYSPROCEDURE.proc_defn
FROM SYS.SYSPROCEDURE
where SYSPROCEDURE.proc_name = 'mt_103_rc';
output to d:\proc.sql format ascii

create procedure mt103rc(in skey varchar(18),out text varchar(32727))
begin
      select distinct ms_text into text from ms_
      where ms_stat = 'R'
      and ms_dirc = 'O'
      and ms_mtyp = '103'
      and ms_dbky like skey;
end

using System;
using System.Data;
using iAnywhere.Data.AsaClient;
using System.Windows.Forms;
using System.Diagnostics;
using System.Collections.Specialized;

namespace ComGenerator
{
      /// <summary>
      /// Summary description for CGetFromTs.
      /// </summary>
      public class CGetFromTs
      {
            private string str_conn;
            private string str_cmd;
            private StringCollection str_rslt;
            //Anywhere data types
            private AsaCommand asa_cmd;
            private AsaConnection asa_conn;
            private AsaDataAdapter asa_adapter;
            //Common data types
            private DataSet ds;
```

```
private DataTable dt;
public CGetFromTs()
{
      //
      str_conn = "Data Source=TS;UID=TS_DBA;PWD=TS_DBA";
      str_cmd = "select * from sys.systable where creator=1";
      //
}
public CGetFromTs(string cmd)
{
      //
      str_conn = "Data Source=TS;UID=TS_DBA;PWD=TS_DBA";
      str_cmd = System.String.Copy(cmd);
      //
}
public bool SetConnection()
{
      bool flag=false;
      try
      {
            asa_conn = new AsaConnection(str_conn);
            asa_conn.Open();
            flag = true;
      }
      catch(AsaException ex )
      {
            MessageBox.Show( ex.Errors[0].Source + " : "
                  + ex.Errors[0].Message + " (" +
                  ex.Errors[0].NativeError.ToString() + ")",
                  "Failed to connect" );
      }
      return flag;
}
public StringCollection Get_103(string str_dbkey)
{
      asa_conn = new AsaConnection(str_conn);
      try
      {
            asa_conn.Open();
            asa_cmd = new AsaCommand("read_103",asa_conn);
            asa_cmd.CommandType = CommandType.StoredProcedure;
```

```
                        asa_cmd.Parameters.Add("@dbky",AsaDbType.Char,18);

                        asa_cmd.Parameters[0].Value = str_dbkey;
                        asa_adapter = new AsaDataAdapter(asa_cmd);
                        AsaDataReader rdr = asa_cmd.ExecuteReader();
                        ds = new DataSet();
                        ds.EnforceConstraints = false;
                        int i = asa_adapter.Fill(ds);
                        ds.EnforceConstraints = true;
                        str_rslt = new StringCollection();
                        while(rdr.Read())
                        {
                              str_rslt.Add(rdr[0] + "#" + rdr[1] + "#"+
rdr[2] + "#"+ rdr[3] + "#"+ rdr[4] + "#"+ rdr[5] + "#"+ rdr[6]);
                        }
                  }
                  catch(AsaException asaex)
                  {
                        LogException(asaex);
                  }
                  catch(Exception ex)
                  {
                        throw ex;
                  }
                  finally
                  {
                        asa_conn.Close();
                  }
                  return str_rslt;
            }
            public string Get_103_all(string str_dbkey)
            {
                  string str_rslt = new string(' ',1500);
                  str_rslt = str_rslt.Trim();
                  asa_conn = new AsaConnection(str_conn);
                  try
                  {
                        asa_conn.Open();
                        asa_cmd = new AsaCommand("rmte",asa_conn);
                        asa_cmd.CommandType = CommandType.StoredProcedure;
                        asa_cmd.Parameters.Add("@dbky",AsaDbType.Char,18);
```

```
                    asa_cmd.Parameters[0].Value = str_dbkey;
                    if((str_rslt = (string)asa_cmd.ExecuteScalar()) ==
null )
                        str_rslt = " ";
                }
                catch(AsaException asaex)
                {
                    LogException(asaex);
                }
                catch(Exception ex)
                {
                    throw ex;
                }
                finally
                {
                    asa_conn.Close();
                }
                str_rslt = str_rslt.Trim();
                return str_rslt;
            }
            public bool CloseCon()
            {
                asa_conn.Close();
                return true;
            }
            private void LogException(AsaException exc)
            {
                EventLog el = new EventLog();
                el.Source = "ComGenerator";
                string strMessage;
                strMessage = "Exception Number : " + exc.Errors.Count
+exc.Message + " has ocurred";
                el.WriteEntry(strMessage);
                foreach(AsaError asaerr in exc.Errors)
                {
                    strMessage = "Message" + asaerr.Message+
                        "Source " + asaerr.Source;
                    el.WriteEntry(strMessage);
                }
            }
            public string GetRow(DataRow row)
```

```
{
      DataTable tbl = row.Table;
      string str = new string(' ',(tbl.Columns.Count)*255);
      str = str.Trim();
      foreach(DataColumn col in tbl.Columns)
      {
            str = str+row[col]+"#";
      }
      return str;
}
public DataSet CreateDataSet()
{
      DataSet ds = new DataSet();
      DataTable tbl;
      DataColumn col;
      ForeignKeyConstraint fk;
      //Add Table mt_103_rcvd in User ds
      tbl = ds.Tables.Add("mt_103_rcvd");
      col = tbl.Columns.Add("id_103",typeof(int));
      col.AutoIncrement = true;
      col.AutoIncrementSeed = -1;
      col.AutoIncrementStep = -1;
      col = tbl.Columns.Add("f_20",typeof(string));
      col.MaxLength = 16;
      col = tbl.Columns.Add("vald",typeof(string));
      col.MaxLength = 8;
      col = tbl.Columns.Add("curc",typeof(string));
      col.MaxLength = 3;
      col = tbl.Columns.Add("amnt",typeof(string));
      col.MaxLength = 17;
      col = tbl.Columns.Add("sndr",typeof(string));
      col.MaxLength = 12;
      col = tbl.Columns.Add("text",typeof(string));
      col.MaxLength = 32767;
      col = tbl.Columns.Add("ms_dbky",typeof(string));
      col.MaxLength = 18;
      col.AllowDBNull = false;
      tbl.PrimaryKey = new DataColumn[] {tbl.Columns[0]};

      //Add Table mt_191_send in User ds
      tbl = ds.Tables.Add("mt_191_send");
```

```
                col = tbl.Columns.Add("id_191",typeof(int));
                col.AutoIncrement = true;
                col.AutoIncrementSeed = -1;
                col.AutoIncrementStep = -1;
                col = tbl.Columns.Add("f_20",typeof(string));
                col.MaxLength = 16;
                col = tbl.Columns.Add("f_21",typeof(string));
                col.MaxLength = 16;
                col = tbl.Columns.Add("vald",typeof(string));
                col.MaxLength = 8;
                col = tbl.Columns.Add("curc",typeof(string));
                col.MaxLength = 3;
                col = tbl.Columns.Add("amnt",typeof(string));
                col.MaxLength = 17;
                col = tbl.Columns.Add("sndr",typeof(string));
                col.MaxLength = 12;
                col = tbl.Columns.Add("stat",typeof(string));

                fk = new
ForeignKeyConstraint(ds.Tables[0].Columns[0],ds.Tables[1].Columns[8]);
                ds.Tables[1].Constraints.Add(fk);
                return ds;
          }
          public bool FillDs(int indTable,string[] strings,int col)
          {
                bool flag = false;
                object[] aValues;
                foreach(string str in strings)
                {
                      aValues = str.Split('#');
                      ds.Tables[indTable].LoadDataRow(aValues,false);
                      flag = true;
                }
                return flag;
          }
     }
}
```

Додаток Б

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій
Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Керівника економічної частини Професора Вагонової О.Г.
(прізвище, ім'я, по батькові, вчене звання)

на магістерську роботу

Студента ІІ курсу групи 121м-20-1 Самарця Назара Олександровича
(прізвище, ім'я, по батькові)

На тему: Розробка програмного забезпечення аналізу особистості на основі зображення відбитків.

«___»_____2022 р. _____
(підпис)

Додаток Д

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файла	**Опис**
Пояснювальні документи	
Самарець.doc	Пояснювальна записка кваліфікаційної роботи. Документ Word.
Самарець.pdf	Пояснювальна записка кваліфікаційної роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація.ppt	Презентація роботи