МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітня програма: «Комп'ютерна графіка і Web-дизайн»

Група: 4КГ-05

Дипломний проект

здобувача освіти денної форми навчання
КГ.05.32.000.ДП

ЧЕЧИКОВОЇ
ЮЛІЇ ПАВЛІВНИ

м. Одеса
2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**
Освітня програма: **«Комп'ютерна графіка і Web-дизайн»**
Група: **4КГ-05**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи) на тему:

Розробка програмних засобів для порівняння папілярних візерунків відбитків пальців

Проектний матеріал складається з пояснювальної записки на 59 сторінках та графічного (презентаційного) матеріалу на 14 аркушах (слайдах).

Дипломник _____ (Чечикова Ю.П.)
Керівник _____ (Кривченко Ю.В.)

Консультанти:
з економічної частини _____ (Копайгородська Т.Г.)
з охорони праці_____ (Чорновол Н.І.)
з дотримання вимог ЄСКД _____ (Петрашова В.І.)
старший консультант _____ (Скорнякова О.В.)

До захисту допущений
Голова циклової комісії _____ (Скорнякова О.В.)
Завідувач відділення _____ (Суліма Ю.Ю.)

Захист «____»_____ 2022 р. Протокол ДКК №_____
Оцінка ДКК _____

Секретар ДКК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення **комп'ютерних систем** Комісія **КТ та ПІ**

Спеціальність **123 «Комп'ютерна інженерія»**

Освітня програма **«Комп'ютерна графіка і Web-дизайн»**

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР_____

Беркань І.В.

"_____"_____2022 р.

З А В Д А Н Н Я

на дипломний проект (роботу)

Здобувачеві (здобувачці) освіти *Чечикової Юлії Павлівні*

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) *Розробка програмних засобів для порівняння папілярних візерунків відбитків пальців*

затверджена наказом по коледжу від "____"_____202___р. №_____

2. Термін здачі закінченого проекту (роботи) _____

3. Вихідні данні до проекту (роботи) _____

1. Реалізувати етапи обробки: виправлення спотворень зображень відбитків пальців; виділення мінюцій; виключення ненадійних мінюцій; конвертація параметрів; облік кореляції зображень; порівняння одного відбитку з іншими у базі даних; 2. Передбачити модульну структура програмного забезпечення; 3. Забезпечити роботу програмного застосунку у операційній системі MS Windows 10 (x86-x64)

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Функціональна структура системи ідентифікації особи; Етапи роботи системи ідентифікації особи; Аналітичний огляд рішень для ідентифікації особи; Розробка та опис функціональної структури системи розпізнавання; Алгоритм перетворення абсолютних параметрів мінюцій до відносних; Алгоритм порівняння структурних представлень відбитків пальців; Розробка керівництва програміста;Розробка керівництва оператора застосунку

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Приклади біометричних систем з розпізнаванням відбитків пальців; Зображення відбитків пальців та аналіз їх візерунків; Область допуску, у якій положення крапки відносно іншої збіглося; Відносні параметри розташування точок; Функціональна схема системи порівняння папілярних візерунків відбитків пальців; Функціональна схема підсистеми розпізнавання; Блок-схеми підпрограм системи розпізнавання відбитків; Інтерфейс програмного застосунку для порівняння відбитків пальців та результати порівняння; Зображення оброблених відбитків пальців з виділеними мінюціями

6. Консультанти по проекту (роботі), із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1.Технологічний	*Кривченко Ю.В.*		
розділ			
2.Екон. частина	*Копайгородська Т.Г.*		
3.Охорона праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		

7. Дата видачі завдання _____

Керівник *Кривченко Ю.В.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту (роботи)	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1.	*Вступ. Постановка задач розробки*	*5.05.2022*	
2.	*Обґрунтування доцільності розробки аналізатору*	*7.05.2022*	
	відбитків пальців		
3.	*Аналітичний огляд рішень для розпізнавання*	*10.05.2022*	
	користувача за відбитками пальців	*14.05.2022*	
4.	*Основні вимоги до системи розпізнавання*	*16.05.2022*	
	користувача		
5.	*Вибір технічних та програмних засобів розробки*	*19.05.2022*	
6.	*Опис вхідної та вихідної інформації системи*	*21.05.2022*	
7.	*Опис алгоритму перетворення параметрів мінюцій*	*23.05.2022*	
8.	*Опис алгоритму порівняння структурних представлень*	*26.05.2022*	
9.	*Складання програми в середовищі Visual Studio C++*	*28.05.2022*	
10.	*Реалізація та опис контрольного прикладу*	*30.05.2022*	
11.	*Створення керівництва програміста та користувача*	*4.06.2022*	
12.	*Аналіз результатів, підготовка презентації*	*10.06.2022*	
13.	*Розробка питань з економіки охорони праці*	*13.06.2022*	

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КГ 05. 32 000. 00 ДП	Дипломний проект		
А4			КГ 05. 32 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КГ 05. 32 000. 00 ДП		
Розробив		Чечикова Ю.П.			Розробка програмних засобів для порівняння папілярних візерунків відбитків пальців	Літ.	Аркуш 4 / 59
Перевірив		Кривченко Ю.В.				Н Д П	
Н. Контр.		Петрашова В.І.				ВСП "ОТФК ОНТУ" гр.4КГ-05	
Затверд.		Скорнякова О.В.					

ЗМІСТ

ВСТУП

Відбиток пальця утворює так звані папілярні лінії на гребенеких виступах шкіри, розділених борозенками. З цих ліній складаються складні узори (дугові, вузлові і завитокові), які володіють властивостями індивідуальності і неповторності, що дозволяє абсолютно надійно ідентифікувати особу. Хоча відсоток відмови в доступі уповноважених користувачів складає близько 3%, відсоток помилкового доступу – менше одного до мільйона. Переваги доступу по відбитку пальця – простота використання, зручність і надійність. Вірогідність помилки при ідентифікації користувача набагато менше порівняно з іншими біометричними методами [2].

Образ відбитку 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 поліпшення зображення. Цей алгоритм забезпечує зворотний зв'язок з датчиком для регулювання параметрів сканування. Коли датчик фіксує остаточний образ, алгоритм налаштовує контрастність і чіткість зображення відбитку для отримання найкращої якості.

Даний дипломний проект присвячено розробці програмних засобів для порівняння папілярних візерунків відбитків пальців.

Методи упізнання відбитку пальця засновані на порівнянні із зразками або на використанні характерних деталей. При упізнанні по деталях з образу витягуються лише специфічні місця, де знайдена особливість (деталь). Вміст шаблону в цьому випадку складають відносні координати і відомості про орієнтацію деталі. Алгоритм розпізнавання відшукує і порівнює між собою відповідні деталі. Ні поворот відбитку пальця, ні його паралельне перенесення (зсув) не впливають на функціонування системи, оскільки алгоритм працює з відносними величинами. Для порівняння на бітовому образі знаходяться локальні особливості папілярного узору – мінуції. Алгоритм має проводити обхід по контуру гребенів для пошуку мінуцій.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

Аналізатор відбитків пальців призначений для ідентифікації особи на основі біометричних параметрів, а саме папілярних візерунків. Система призначена для обробки графічних зображень відбитків. Система дозволяє порівняти декілька відбитків один з одним по виділеним локальним особливостям – мінуціям і їх відносним параметрам (розташуванню одних мінуцій щодо всіх інших), що гарантує незалежність порівняння від паралельного перенесення і обертання.

Програмний продукт зможе знайти вживання в різних прикладних системах, включаючи:

1) системи цивільної ідентифікації;
2) криміналістичні системи ідентифікації;
3) великомасштабні комерційні додатки.

Системи цивільної ідентифікації включають:

- паспорти водіїв;
- національні ідентифікаційні карти громадян;
- реєстрація виборців;
- реєстрація для соціальних програм;
- імміграційна реєстрація, візи;
- ідентифікація співробітників державних установ.

Криміналістичні системи ідентифікації включають:

- визначення, чи знаходиться даний громадянин в розшуку;
- визначення наявності колишніх судимостей;
- реєстрація в'язнів/контроль доступу;
- мобільні і віддалені додатки;
- обробка відбитків пальців, отриманих з місць злочину.

Великомасштабні комерційні додатки включають:

- доступ до Web-ресурсів, електронна комерція;
- доступ для користувачів і співробітників;
- фінансові сервіси, перевірка оплати;

- доступ в будівлі і приміщення;
- програми лояльності.

1.1 Функціональна структура системи ідентифікації особи

Функціональна схема системи, призначеної для ідентифікації користувачів на основі відбитків пальців, приведена на рис. 1.1.

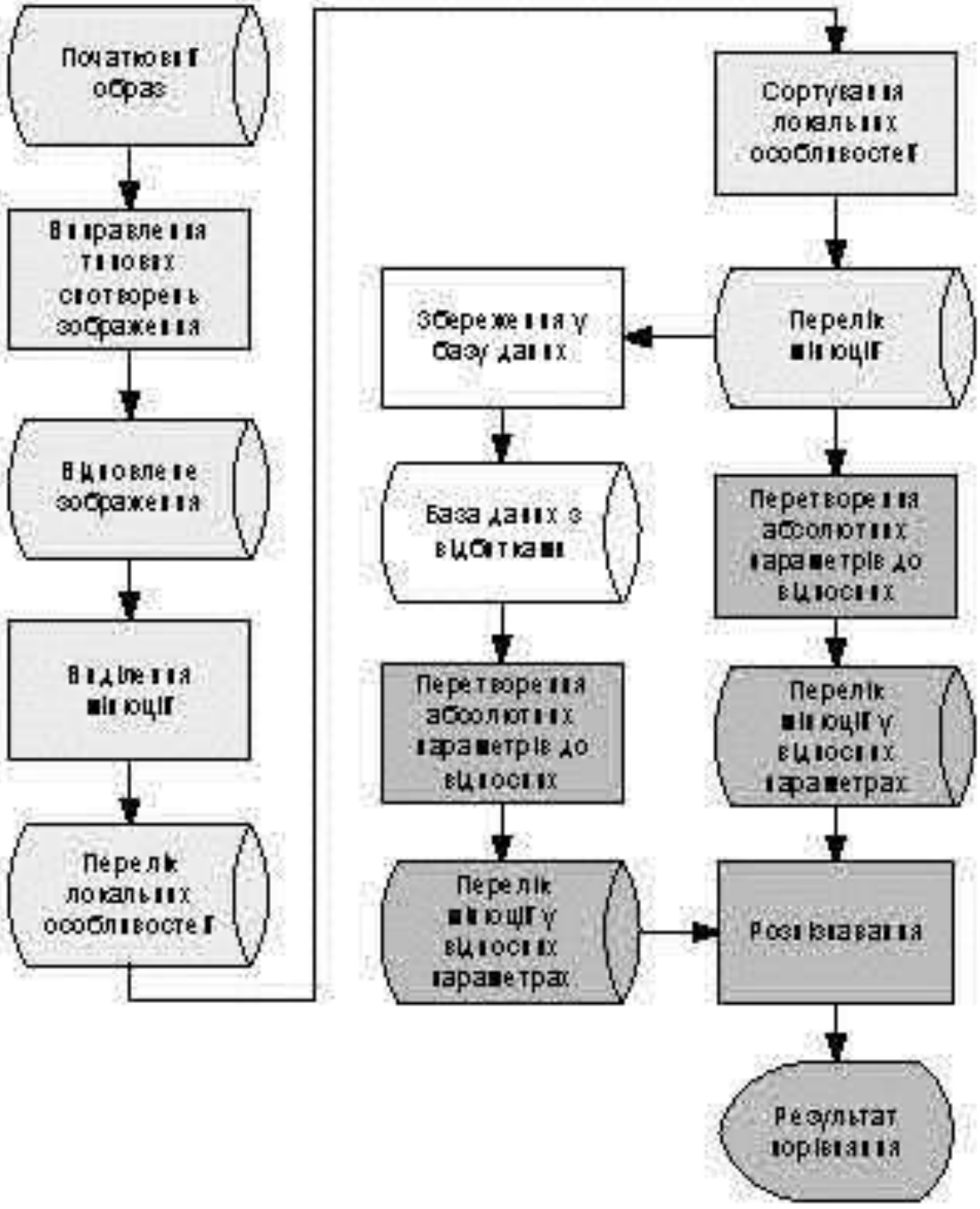

Рисунок 1.1. Функціональна схема системи ідентифікації особи

На рис.1.1 зеленим кольором показана підсистема аналізу, а синім кольором – підсистема розпізнавання. Обробка зображень складається з наступних етапів:

1) введення зображення в систему ідентифікації особи по відбитках пальців;
2) у підсистемі аналізу зображення відбувається обробка растру з метою придушення шумів, а також усунення типових спотворень зображення, таких як розриви або злипання дуг папілярного візерунку;
3) у підсистемі аналізу зображення відбувається виділення локальних особливостей, таких як закінчення і роздвоєння, для подальшого розпізнавання відбитку;
4) у підсистемі розпізнавання відбувається перетворення абсолютних параметрів спеціальних крапок до відносних параметрів для запобігання впливу паралельного перенесення і повороту пальця при скануванні відбитку;
5) розпізнавання відбитку по базі з існуючими відбитками відбувається на основі відносних параметрів кожної крапки для кожного відбитку, що зберігається в базі.

1.2 Обґрунтування мети створення ідентифікації особи

На даний момент надійний інформаційний захист є одним з основних критеріїв, по яким мають відбиратися системи, призначені для зберігання і обробки важливої інформації. Це обумовлено існуючою вірогідністю несанкціонованого доступу в такі системи, оскільки вони мають широку інформаційну взаємодію з суміжними системами управління через мережу INTRANET. Тому забезпечення інформаційної безпеки має бути найважливішим етапом при їх розробці.

Захист на основі біометричних параметрів людського тіла, зокрема по відбитку пальця, володіє рядом безперечних плюсів: 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 відбитку пальця для ідентифікації особи є найбільш зручним зі всіх біометричних методів. Вірогідність помилки при ідентифікації користувача в такий спосіб набагато менше порівняно з іншими біометричними методами. Крім того, пристрій ідентифікації по відбитку пальця не вимагає багато місця на клавіатурі або в механізмі.

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

Хоча на ринку існують готові системи, але на ряду зі своїми перевагами вони володіють рядом недоліків, таких як закритість початкового коду і алгоритму, а також висока ціна. Унаслідок цього є сенс в розробці системи, яка б надавала можливість розробникам мати готову базу для розробки власних проектів на основі біометричних технологій.

Метою даної роботи, зокрема, є реалізація такого перетворення зображення, при якому дані об розташування унікальних особливостей зберігаються якнайповніше і з найменшим вмістом помилкової інформації.

Дана робота направлена на полегшення розробки алгоритмів обробки зображень, спрощення аналізу експериментальних даних і виявлення спільних закономірностей.

1.3 Етапи роботи системи ідентифікації особи

Реалізація системи ідентифікації особи по папілярному візерунку дозволить інтегрувати в єдиному інтерфейсі всі етапи обробки зображення відбитку пальця і порівняння його з іншими відбитками:

1) аналіз параметрів зображення, виявлення дефектів сканування і їх

усунення;

2) виділення локальних особливостей – мінуцій. Формування списку мінуцій в абсолютних параметрах;
3) сортування списку абсолютних параметрів, виключення помилкових і ненадійних мінуцій;
4) конвертація абсолютних параметрів у відносні, формування списку відносних параметрів;
5) установка системи допусків для обліку кореляції зображень;
6) облік всіх зареєстрованих відбитків;
7) розпізнавання відбитку за допомогою пошуку співпадаючих спеціальних крапок в базі з відбитками;
8) спосіб зберігання опису відбитків дозволяє застосовувати результат роботи програми для різних сфер діяльності

1.4 Аналітичний огляд рішень для ідентифікації особи

Метод упізнання особи по відбитках пальців відомий досить давно і з появою електронно-обчислювальної техніки почали з'являтися програмні та апаратні рішення для аналізу і порівняння зображень.

Система ідентифікації особи по відбитках пальців призначена для роботи з цифровими зображеннями, отриманими за допомогою сканування. Отримання електронного представлення відбитків пальців з добре помітним папілярним візерунком – достатньо складне завдання.

На сьогоднішній момент можна виділити наступні види сканерів відбитків пальців по використовуваним фізичним принципам:

- оптичні;
- кремнієві;
- ультразвукові.

Досить старою технологією сканування відбитку є оптична. Сканування відбитку пальця мініатюрними на ПЗЗ або КМОП-чипі дозволило істотно зменшити вартість систем ідентифікації. Але цей спосіб зняття відбитку стикається з деякими важко вирішуваними проблемами: отримуваний образ

залежить від навколишнього освітлення, на кордонах образу можливі спотворення, датчик може бути відносне легко "обдурений" (деякі дешеві датчики можна "обдурювати" друкованою копією). Залишаються проблеми і з розмірами сканера. Датчик не може бути менше, ніж фокусна відстань камери. Серед головних переваг оптичних систем – відносно низька ціна і практична нерозливість до дії електростатичного розряду.

Відносно новою є технологія використання електромагнітного поля. Датчик випромінює слабкий електромагнітний сигнал, який слідує по гребенях і западинах відбитку пальця і враховує зміни цього сигналу для складання образу відбитку. Такий принцип сканування дозволяє проглядати малюнок шкіри під шаром омертвілих клітин, що приводить до добрих результатів при розпізнаванні блідих або стертих відбитків. Залишається проблема відсутності прийнятного співвідношення між розміром датчика і його роздільною здатністю.

Ще одна перспективна технологія – ультразвукова. Тривимірний ультразвуковий сканер вимірює претнуту поверхню пальця свого роду радаром. Цей метод сканування може бути особливо зручний, наприклад, в охороні здоров'я. Він не вимагає торкання яких-небудь зчитуючих пристроїв датчика стерильними руками, а відбиток легко зчитується навіть через гумові або пластикові рукавички хірурга. Головна незручність ультразвукової технології – її висока вартість і тривалий час сканування.

Розглянемо найбільш характерні приклади використання розглянутих вище технологій для розпізнавання користувачів по папілярному візерунку.

1.4.1 Біометричний термінал BioLink

BioLink, провідний постачальник технологій забезпечення безпеки, проектує, проводить і продає передові біометричні продукти, засновані на принципі дактилоскопії. Пропоновані рішення складають основу для систем аутентифікації користувачів в комп'ютерних мережах, платформах електронної комерції і системах забезпечення безпеки фізичного доступу. BioLink пропонує гамму продуктів, заснованих на фірмових технологіях сканування відбитків пальців і обробки зображень, а також на алгоритмі ідентифікації «один до

багатьох», вирішуючих багато існуючих сьогодні проблем безпеки.

BioLink BioTime – це система управління робочим часом, що є новітньою розробкою компанії BioLink.

Рисунок 1.2. Біометричний термінал BioLink FingerPass TM

Система BioTime та біометричний термінал BioLink FingerPass (рис.1.2) спрощує звичайні задачі обліку і управління робочим часом і забезпечує простоту, легкість і зручність реєстрації приходу і відходу співробітників компанії. Крім того, система BioTime надає різні види звітів по запізненнях, недоробках і переробках співробітників, часі їх приходу і відходу, а також автоматизує створення табеля обліку робочого часу.

Програмний сервер BioLink Authenteon Software Appliance (ASA) – це програмне забезпечення для порівняння шаблонів відбитків пальців BioLink. ASA об'єднує в собі паролинний захист і клієнт-серверну аутентифікацію при вході в Windows, Novell і NFS при вирішенні однієї з самих актуальний на сьогоднішний день проблем захисту – позитивної ідентифікації користувачів корпоративноїмережі. Сервер підтримує до 300 користувачів.

1.4.2 Клавіатура Microsoft IntelliMouse Explorer Fingerprint Reader

Лінійка продуктів Microsoft з використанням біометричних технологій – сканер відбитків пальців, клавіатура з вбудованим сканером (рис.1.3) і бездротова оптична миша з сканером володіє можливостями:

1) зняття відбитку пальця при короткочасному доторканні пальця до сканера;
2) ведення менеджера паролів для веб-серверних інтерфейсів;
3) можливість ідентифікації особи для входу в систему одним торканням пальця.

Рисунок 1.3. Клавіатура Microsoft із вбудованим сканером відбитків пальців

Продукт підходить для використання з особистим ПК. Програмне забезпечення має дуже обмежену функціональність. Немає можливості отримати параметри відсканованого відбитку пальця, встановити додаткові дії в залежності від того, який палець був прикладений.

1.4.3 Смартфон з функцією розпізнавання відбитків пальців Samsung Galaxy S20

Смартфони S10 та S20 оснащені ультразвуковим сканером відбитків пальців, який розташований на передній частині екрану (рис.1.4). Відбитки пальців використовуються як для набору номера, так і для використання у мультимедійних додатках. При включенні телефону відбувається сканування відбитку пальця. У разі трикратної невдачі при розпізнаванні відбитку пальця пропонується ввести пароль. Сканер може погано розпізнати пальці, якщо вони занадто вологі або занадто сухі, наприклад, в дуже холодних умовах. Треба натиснути на екран так, щоб кінчик пальця поширювався по широкій поверхні області розпізнавання відбитків пальців. Ультразвуковий сканер відбитків пальців знаходиться під дисплеєм.

Рисунок 1.4. Смартфон Samsung Galaxy S20
із вбудованим сканером відбитків пальців

1.4.4 Флеш-накопичувач із USB-сканером відбитків пальців

Існують області діяльності, в яких конфіденційність інформації вкрай важлива. Тому час від часу з'являться продукти з посиленими системами захисту інформації – шифруванням і рівними способами персонального доступу до даних. Саме такий флеш-накопичувач випустила компанія Chinavasion (рис.1.5). Пристрій має на корпусі сканер відбитків пальців, так що можливість використання накопичувача сторонніми особами повністю виключена.

Рисунок 1.5. Флеш-накопичувач із вбудованим сканером відбитків пальців

Накопичувач здатний тримати в пам'яті до 10 різних відбитків (по одному на кожен палець), а також звичайний пароль доступу. Корисною додатковою функцією пристрою є можливість запуску комп'ютера за допомогою сканування відбитку пальця.

1.5 Основні вимоги до системи ідентифікації особи

1.5.1 Критерії ефективності функціонування системи

Створення системи розпізнавання особи дозволить отримати нову можливість по підготовці зображень до структурного аналізу, розробити інструмент, поліпшуючий якість графічної інформації за рахунок зниження спотворень і шумів. Для оцінки ефективності роботи системи можна використовувати якість отримуваних на виході зображень і їх структурний опис, а також рівень правильного розпізнавання відбитків, про який можна судити по кількості відмов для правильного відбитку, і кількості входів для невірного відбитку. Розроблена система буде мати відкритий код, дозволить отримувати структурний опис папілярного візерунку і його порівняння з іншими. Алгоритм має підходити для роботи не лише із зображеннями відбитків пальців, але і для інших бітових зображень, таких як символьна інформація, шрифти і підписи.

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

1) модифікація зображення, виправлення спотворень;
2) виділення локальних особливостей – мінуцій, формування списку мінуцій в абсолютних параметрах;
3) сортування списку абсолютних параметрів, виключення помилкових і ненадійних мінуцій;
4) конвертація абсолютних параметрів у відносні, формування списку відносних параметрів;
5) установка системи допусків для обліку кореляції зображень;
6) порівняння одного відбитку з безліччю інших.

1.5.2 Вимоги до структури програмного забезпечення

Побудова системи ідентифікації особи по відбитках пальців передбачає модульну структуру. Спільний інтерфейс і можливість доступу до всіх модулів у складі системи повинна забезпечувати оболонка. З оболонки мають викликатися наступні модулі: підсистема аналізу зображення, підсистема порівняння одного відбитку з безліччю інших. Обмін даними між підсистемами має відбуватися через проект в рамках спільної оболонки. Підсистема аналізу зображення повинна забезпечувати можливість отримання основних статистичних характеристик папілярного візерунку по ключових ділянках. Підсистема передбачає наявність засобів для отримання якісного образу відбитку пальця. Підсистема порівняння зображень відбитків служить для автоматизованого виявлення схожості різних зображень папілярного візерунку.

Система призначена для обробки бітових зображень. Унаслідок неточностей, шумів і апроксимацій, що вносяться устаткуванням (сканер або будь-який інший дискретизуючий графіку пристрій) в зображенні з'являються шуми різної природи. Система має дозволяти частково позбавитися від цих спотворень. Основним видом інформації, що обробляється в системі ідентифікації особи, є графічна інформація в растровому представленні. Такий вид даних сприймається людиною безпосередньо і цілісно, тому необхідно забезпечити засоби наочної візуалізації зображень на різних етапах обробки.

1.5.3 Вимоги до програмних засобів розробки

Для реалізації і функціонування проекту необхідне загальносистемне програмне забезпечення: ОС Windows 7-10, в основі яких лежить ядро, що характеризується 32- або 64- розрядною обчислювальною архітектурою і повністю захищеною моделлю пам'яті, що забезпечує надійне обчислювальне середовище.

Розробка програмних засобів для порівняння папілярних візерунків відбитків пальців вестиметься з використанням середовища для розробки додатків Microsoft Visual Studio C++ 2019. Середовище розробки включає

високопродуктивний 32- і 64-бітовий компілятор, що дозволяє оптимізувати створюваний код. Microsoft Visual Studio C++ включає великий набір засобів, які підвищують продуктивність роботи програмістів і скорочують тривалість циклу розробки. Багатофункціональне інтегроване середовище розробки Microsoft Visual Studio C++ 2019 включає компілятор, що задовольняє стандарту ANSI/ISO, вбудований дизайнер форм, багатий набір засобів для роботи з компонентами, інструмент Solution Explorer, менеджер проектів і відладчик. Зручність розробки і ефективність створених в даному середовищі розробки програм роблять Microsoft Visual Studio C++ оптимальним вибором для побудови системи розпізнавання користувача.

1.6 Розробка та опис функціональної структури системи розпізнавання

При торканні пальця до скануючого пристрою можливий зсув або поворот зображення відбитку пальця в порівнянні з тим, що вже зберігається в базі. Дані погрішності не повинні робити вплив на результат розпізнавання отримуваного відбитку пальця. Для цього був розроблений алгоритм перетворення абсолютних параметрів мінюцій до відносних. Завдяки такому перетворенню удається запобігти негативному впливу повороту і зсуву, і розпізнати відбиток, навіть якщо він обернений на 180°.

Представимо структурне представлення відбитку пальців у вигляді списку M, що містить параметри спеціальних крапок. Кожен з наборів параметрів є однією крапкою. Для приведення параметрів до відносних параметрів необхідно провести огляд і перетворення всіх крапок. Унаслідок еластичності шкіри і зростання людини відстань між крапками може змінитися, також крапки, що знаходяться ближче до краю відбитку можуть зміститися відносно інших крапок, що не повинне впливати на результат розпізнавання, проте різні крапки також не мають бути прийняті за одну. Для цього в підсистемі розпізнавання була розроблена система допусків при порівнянні двох відбитків. Значення допусків були набуті дослідним шляхом, так само як і умови збігу відбитків.

Таким чином, задача розпізнавання відбитку пальців по абсолютних

параметрах мінюцій на зображенні може бути розбита на декілька підзадач:

1) розробка алгоритму, що забезпечує компенсацію впливу переміщення або повороту відбитку пальця;
2) розробка і реалізація системи допусків і критеріїв схожості при пошуку відповідного відбитку по існуючій базі відбитків;
3) порівняння мінюцій на відбитках пальців з використанням отриманих критеріїв і виявлення схожості відбитків за кількістю мінюцій, що збіглися.

1.6.1 Вхідна інформація для системи розпізнавання

Вхідною інформацією є список мінюцій в абсолютних параметрах, розташований в пам'яті, який містить всі необхідні параметри. Кожен елемент масиву містить всі необхідні параметри мінюцій: координати цілого типу – 2х4 байта, кут напрямку 8 байт, тип крапки 1 байт, тому спільний розмір масиву має бути кратний 2*4+8+1 = 17 байт.

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

де $X_i, Y_i \ (i \in \overline{1,k})$ – координати мінюцій на растровому представленні зображення відбитку пальців, цілі числа, величина яких обмежена розміром зображення відбитку у пікселах;

$\alpha_i \ (i \in \overline{1,k})$ – напрямок передбачуваного продовження гребеня на відбитку пальців в точки типа закінчення і направлення злипання для крапки типа роздвоєння, дробове число, величина якого змінюється (–pi +pi);

$T_i \ (i \in \overline{1,k})$ – тип виявленої крапки, бітове поле, набуває 2 значення «роздвоєння» = 0 (false) і «закінчення» = 1 (true);

k – кількість мінюцій на досліджуваному відбитку.

Також у якості вхідної інформації для даної задачі є матриці, що містять список мінюцій в абсолютних параметрах, які зберігаються в базі даних відбитків та містять всі необхідні для розпізнавання параметри, знайдені при

попередніх розрахунках параметрів відбитків пальців. У кожному рядку масиву міститься опис розташування однієї крапки на зображенні відбитку, а також її напрямок, тип і видимість. У таблиці 1.1 приведений формат елементу матриці.

Таблиця 1.1

Поле	Формат	Опис
x	Ціле	Абсциса мінуції на растрі
y	Ціле	Ордината мінуції на растрі
alpha	Ціле	Орієнтація мінуції на растрі
type	Байт	Тип мінуції. Роздвоєння або закінчення
show	Байт	Видимість крапки

1.6.2 Вихідна інформація системи розпізнавання

Вихідною інформацією для даної підсистеми є список, що містить відбитки з бази даних, в яких була виявлена схожість з оброблюваним відбитком. Значення, що говорить про те, що відбитки збіглися, також набуте дослідним шляхом.

У кожному рядку масиву списку міститься опис відбитку, його ім'я, кількість крапок, що збіглися при розпізнаванні і ступінь схожості. У таблиці 1.2 приведений формат рядка.

Таблиця 1.2

Поле	Формат	Опис
Name	Рядкове	Ім'я відбитку (найменування файлу, з якого були узяті параметри)
Count	Ціле	Кількість мінуцій відбитків, що збіглися при порівнянні
Pct	Дріб	Ступінь схожості відбитків у відсотках, набуває значень (0, 100].

1.6.3 Математичне рішення задачі розпізнавання

Перетворення відносних параметрів до абсолютних параметрів компенсує вплив паралельного перенесення і повороту відбитків пальців при скануванні. Перетворення відбувається для кожної виявленої мінуції щодо всієї решті мінуцій на зображенні. На рис. 1.6 представлено зображення відбитку пальця з виявленими на ньому мінуціями.

Рисунок 1.6. Прямий відбиток

Лініями представлено відносне розташування крапок щодо центральної. На рис. 1.7 представлено зображення відбитку того ж пальця, але повернутого відносного першого на 45 градусів. Лініями представлено відносне розташування крапок щодо центральної.

Рисунок 1.7. Повернений відбиток

Для визначення ступеня схожості обробленого відбитку з відбитком, що зберігається в базі, була розроблена система допусків і критеріїв схожості відбитків. Унаслідок еластичності шкіри і нещільного притиснення пальця при знятті відбитку деякі мінюції на новому відбитку можуть зміститися відносно

мінюцій, що зберігаються в базі. Таким чином, крапка вважається за ту, що збіглася, якщо її місце розташування щодо іншої крапок входить в певну область довкола первинного положення. На рис. 1.8 представлена область, в якій положення крапки відносно іншої збіглося.

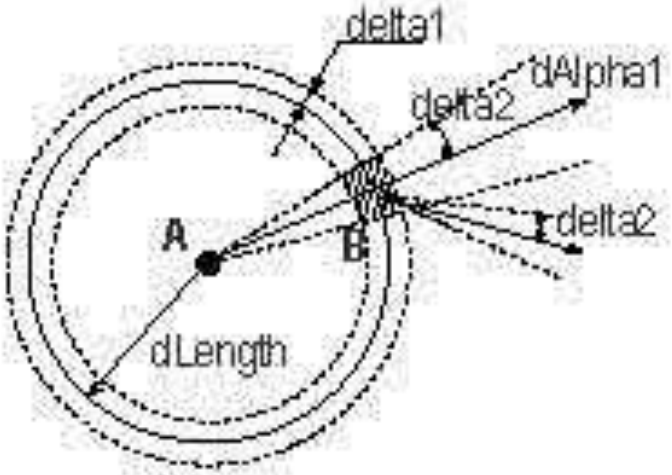

Рисунок 1.8. Область допуску

Для обчислення ступеня схожості двох відбитків відбувається порівняння кожної мінюції на обох відбитках. Список мінюцій, що збіглися, виходить за допомогою відсівання з первинного списку тих мінюцій, які не були виявлені в другому списку, тобто довкола даної крапки не було виявлено жодної іншої крапки, що потрапила в область допуску.

Відбитки вважаються за схожі, якщо кількість крапок, що збіглися, перевищує певний поріг схожості.

Робота підсистеми реалізується наступними етапами:

- перетворення абсолютних параметрів мінюцій до відносних параметрів;
- розробка системи допусків для порівняння мінюцій;
- порівняння структурних представлень відбитків пальців.

Функціональна схема підсистеми представлена на рис.19.

1.7 Алгоритм перетворення абсолютних параметрів мінюцій до відносних

При реалізації даного алгоритму використовується масив інформації, сформований на попередньому етапі обробки зображення відбитку пальця в підсистемі аналізу зображення. Для обробки масивів використовується

двонаправлений список. Списком є вектор: $S = |s_1, s_2, \ldots s_n|$. У дану інформаційну структуру можна заносити і витягувати елементи з будь-якого боку.

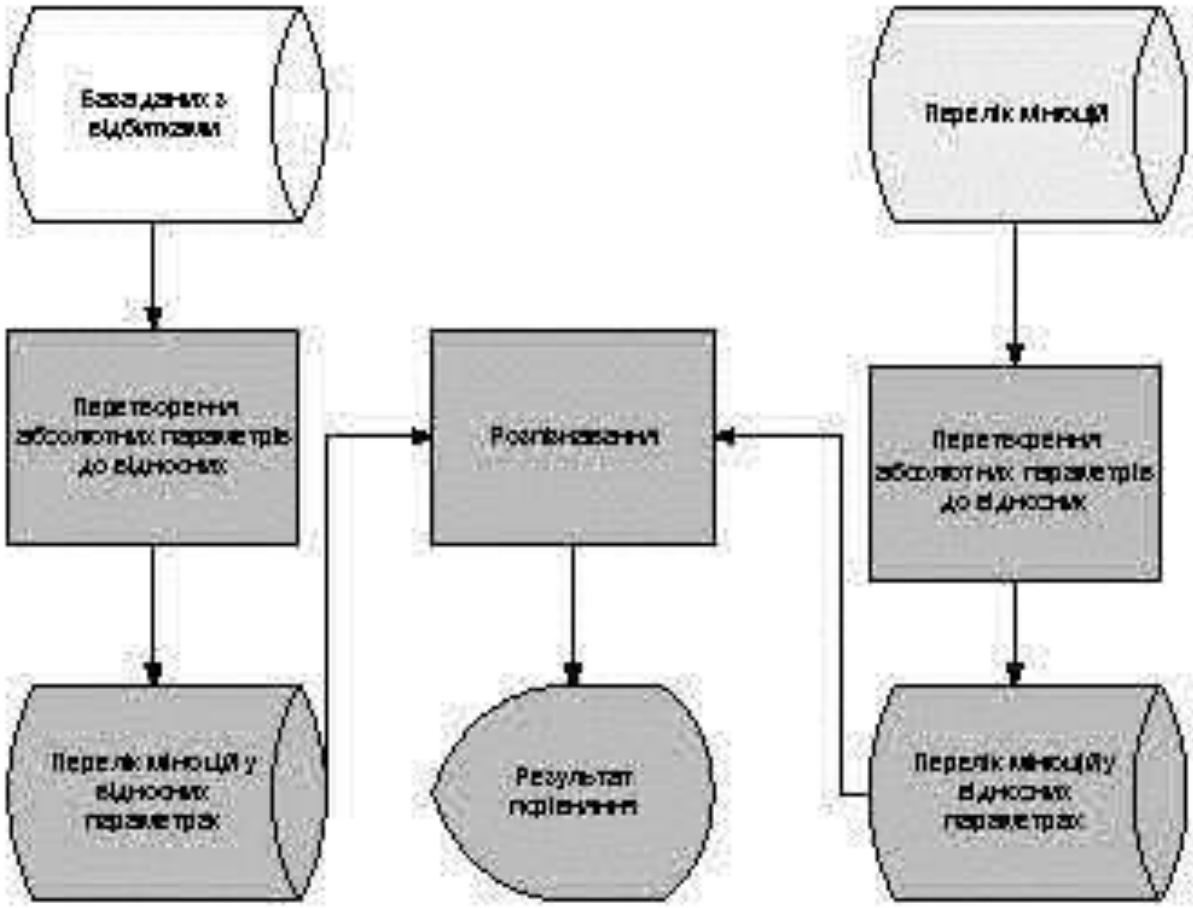

Рисунок 1.9. Функціональна схема підсистеми розпізнавання

Вихідною інформацією для даного завдання є масив розмірністю (k x k) з порожніми діагональними елементами, що містить списки мінуцій і їх взаємне розташування один з одним.

Таблиця 1.3

Поле	Формат	Опис
l_{ij}	Ціле	Відстань між i і j крапками
$A1_{ij}$	Дробове	Кут між власним напрямком точки та її напрямком з точки і у точку j. $A1_{ij} \in [0, 2*M\_PI]$
$A2_i$	Дробове	Кут між власними напрямками точок і і j. $A2_{ij} \in [0, 2*M\_PI]$

Кожен елемент масиву містить опис крапки, відстань до другої крапки, кут між власним напрямком крапки і напрямком у іншу крапку, кут між власними напрямками двох крапок. У таблиці 1.3 наведений формат елементу масиву запису.

Узагальнений математичний опис перетворення наведений вище.

Перетворення відбувається для кожної виявленої мінюції щодо решти всіх крапок по наступних формулах:

$$dLength_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$
$$dAlpha1_{ij} = alpha_{ij} - alpha_i$$
$$dAlpha2_{ij} = alpha_{ji} - alpha_i$$

де i, j – мінюції,

$dLength_{ij}$ – відстань між точками i і j,

$dAlpha1_{ij}$ – кут між напрямком точки i та напрямком на точку j,

$dAlpha2_{ij}$ – кут між напрямком точки i та точки j,

$Alpha_i$ – кут вектора самої крапки,

$Alpha_{ij}$ – кут вектора напрямку від точки i до точки j.

На рис. 1.10 представлено розташування точки i щодо точки j зі всіма отриманими параметрами.

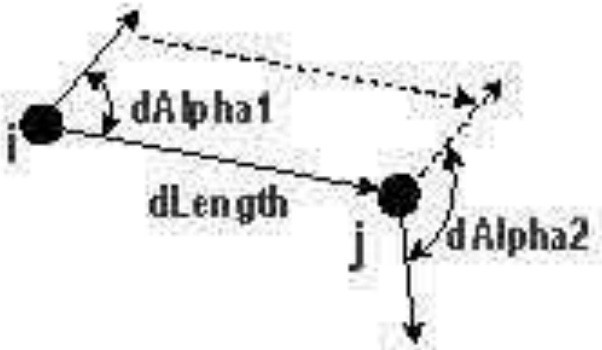

Рисунок 1.10. Відносні параметри

Алгоритм знаходження габаритних розмірів і кількості крапок в безперервній області є таким:

1. Очистити список RelFing з відносними параметрами відбитку.
2. Якщо список AbsFing порожній, перейти до пункту 20.
3. Для кожного елементу IterA1 списку AbsFing виконати пункти 4 – 19.

4. Очистити список listDots з відносними параметрами кромки.
5. Для кожного елементу iterA2 списку AbsFing виконати пункти 5 – 17.
6. Якщо iterA2 == iterA1, перейти до пункту 5.
7. l = |GetS(iterA1->coord, iterA2->coord)|.
8. vecAB = GetAlpha(iterA2->coord, iterA1->coord).
9. tmpa = iterA1->alpha - vecAB.
10. Якщо (tmpa < 0), перехід до п. 11, інакше перехід до п. 12.
11. tmpa = 2*M_PI + tmpa.
12. a1 = |tmpa * 180.0/M_PI + 0.5|.
13. tmpa = iterA2->alpha – iterA1->alpha.
14. Якщо (tmpa < 0), перехід до п. 15, інакше перехід до п. 16.
15. tmpa = 2*M_PI + tmpa.
16. a2 = |tmpa * 180.0/M_PI + 0.5|.
17. Додати в список listDots параметри чергової кромки – l, a1, a2, перейти до п. 5.
18. Відсортувати список listDots.
19. Занести відносні параметри точки listDots в список відбитку RelFing, перейти до п. 3.
20. Кінець.

У наведеному алгоритмі та використовуються наступні умовні позначення:

RelFing – список мінуцій у відносних параметрах;

AbsFing – список мінуцій в абсолютних параметрах;

listDots – відносні параметри кромки;

iterA1 – досліджувана кромка в абсолютних параметрах;

iterA2 – кромка в абсолютних параметрах, відносно якої обчислюється точка iterA1;

l – відстань між точками iterA1 і iterA2;

GETS – функція обчислення відстані;

|| – округлення до найближчого цілого;

vecAB – вектор між напрямками точки iterA1 і iterA2;

GetAlpha – функція обчислення кута між векторами;

a1 = кут між напрямком самої крапки і напрямком на іншу крапку;

a2 = кут між напрямками крапок.

1.8 Алгоритм порівняння структурних представлень відбитків пальців

Даний алгоритм призначений для порівняння двох відбитків, один з яких виходить на попередньому етапі, а другий зчитується з файлу бази даних. Обидва відбитки представлено у вигляді відносних параметрів.

При реалізації алгоритму використовуються відносні параметри мінуцій, отримані в результаті перетворення абсолютних параметрів мінуцій до відносних, що описано вище. Інформація про необхідні параметри мінуцій міститься в списку fmg, кожна запис якого має структуру, приведену у табл. 1.3.

Результатами рішення є список відбитків з бази даних, структурне представлення яких збігається з досліджуваним відбитком або схоже з ним.

Для обчислення ступеня схожості двох відбитків відбувається порівняння кожної мінуції на обох відбитках:

M – список мінуцій на обробленому образі,

k – кількість мінуцій на обробленому образі,

M = {m1, m2 ., mk},

N – список мінуцій одного відбитку з бази відбитків,

l – кількість мінуцій на відбитку з бази відбитків,

N = {n1, n2 ., nl}.

Список мінуцій, що збіглися, виходить за допомогою відсівання з первинного списку тих мінуцій, які не були виявлені в другому списку:

S = {mi, де i=(1.k), P(mi, N)}

r = | S | – кількість крапок, що збіглися.

P(mi, N) – вважається, що крапка збіглася, якщо відносно неї знайдено необхідну кількість задовольняючих умові крапок.

Відбитки вважаються за схожі, якщо кількість крапок, що збіглися, перевищує поріг схожості (p):

$r \geq p$ – умови збігу,

$r < p$ – умова незбігу.

Алгоритм знаходження статистичних характеристик колірного кластера є таким:

1. CONFIRM_VAL = 9; DELTA_L = 10.0; DELTA_A = 10.0; confirmDot = 0; shortneedVal = |min(this->size(),fng.size())/3.0 +0.5|;
2. Для кожної крапки на вхідному tekFing відбитку виконати пункти 3-14;
3. Для кожної крапки на відбитку з бази baseFing виконати пункти 4-14;
4. confirmVal = 0;
5. Для кожної відносної точки tekIter для точки tekFing виконати пункти 6-13;
6. Пошук першої відповідної по відстані крапки в списку baseFing для tekIter;
7. Якщо крапка не знайдена, перейти до п. 5;
8. Для кожної відносної точки baseIter для крапки baseFing виконати п. 9;
9. Якщо точка baseIter не задовольняє умовам з погрішністю, то перехід до п. 8;
10. confirmVal = confirmVal + 1;
11. Якщо confirmVal <= needVal, то перехід до п. 5;
12. Видалити точку baseFing з подальшого перебору, оскільки вона вже збіглася;
13. confirmDot = confirmDot + 1; перехід до п. 3;
14. Кінець.

У наведеному алгоритмі та використовуються наступні умовні позначення:

confirmVal – кількість зв'язаних СТ, що збіглися з поточною СТ;

confirmDot – кількість СТ, що збіглися (спеціальних крапок);

min – функція з 2 вхідними параметрами, результатом якої є мінімальне з вхідних значень;

CONFIRM_VAL = 9;

DELTA_L = 10.0;

DELTA_A = 10.0;

|| – округлення до найближчого цілого;

tekFing – список крапок у відносних параметрах на вхідному відбитку;

baseFing – список крапок у відносних параметрах на відбитку з бази;

tekIter – список крапок відносно досліджуваної (її відносні параметри) для крапки на вхідному відбитку;

baseIter список крапок відносно досліджуваної (її відносні параметри) для крапки відбитку з бази.

1.9 Розробка підпрограми «OnBnClicked Compare»

Підпрограма OnBnClickedCompare призначена для обробки події на діалоговому вікні – натиснення кнопки «Порівняти». Підпрограма проводить перевірку, чи був проведений аналіз відбитку пальця. Результат пошуку зберігається у файл. Текст підпрограми приведений у додатку. Підпрограма OnBnClickedCompare призначена для:

1) перевірки вхідного відбитку на аналіз;
2) збереження результату у файл звіту.

Структура TCompareFing призначена для зберігання інформації про порівнювані відбитки пальців. Структура TCompareFing оголошена таким чином:

```
struct TCompareFing
{
double val;
short cDot;
short nfng;
CString name;
list<TPairAbsDot> dots;
list<TPair3ur> surdots;
};
```

де: val – рівень схожості відбитків,

cDot – кількість крапок, що збіглися,

nfng – номер відбитку,

name – файл відбитку,

dots – крапки, що збігливсяна відбитках,

surdots – оточення на однакових відбитках.

Структура TAbsFing – список крапок в абсолютних параметрах, отриманий у результаті роботи підсистеми аналізу. Структура TAbsFing оголошена таким чином:

```
class TAbsDot
{
public:
CPoint coord;
double alpha;
booltype;
boolshow;
};
```

де: coord – координати,

alpha – напрямок в крапці,

type – тип крапки (1– закінчення, 0– роздвоєння),

show – видимість крапки (1– видима, 0 – прихована).

Вхідні дані для даної підпрограми представлені:

TAbsFing fing – список крапок в абсолютних параметрах, отриманий в результаті роботи підсистеми аналізу. Кожен елемент списку містить всі необхідні параметри для обробки і перетворення в підсистемі розпізнавання. Результат роботи підпрограми зберігається у файл звіт

У даній підпрограмі використовуються наступні підпрограми:

1. PrintReport – виведення результату пошуку у файл звіту;
2. CompareWithBase – функція порівняння поточного відбитку з відбитками з бази даних;
3. Convert – перетворення абсолютних параметрів до відносних.

Схема підпрограми «OnBnClickedCompare» приведена на рис. 1.11.

1.10 Розробка підпрограми «Convert»

Підпрограма Convert призначена для реалізації алгоритму перетворення відбитку з абсолютних параметрів до відносних. Підпрограма порівнює кожну крапку на вхідному відбитку з рештою всіх крапок на відбитку.

Результат перетворення повертається як вихідний параметр функції.

Підпрограма Convert виконує:

1) перетворення відбитку з абсолютних параметрів до відносних;
2) відсортування списку відносних параметрів для збільшення швидкості розпізнавання.

Структура TRelFing призначена для зберігання інформації про відбитки пальців у відносних параметрах. Структура TRelFing оголошена таким чином:

```
class TRelFing: public list<listTRelDot>
typedef list<TRelDot> listTRelDot;
class TRelDot
{
public:
short l,a1,a2;
TAbsDot absDot;
}
```

де: l - відстань між крапками,

a1 – кут між напрямком крапки A і напрямком A->B,

a2 – кут між напрямком крапки B і напрямком A,

absDot – абсолютні параметри (необхідні для відображення на екрані крапок, що збіглися),

TAbsFing fing – список крапок в абсолютних параметрах, отриманий в результаті роботи підсистеми аналізу.

Кожен елемент списку містить всі необхідні параметри для обробки і перетворення в підсистемі розпізнавання:

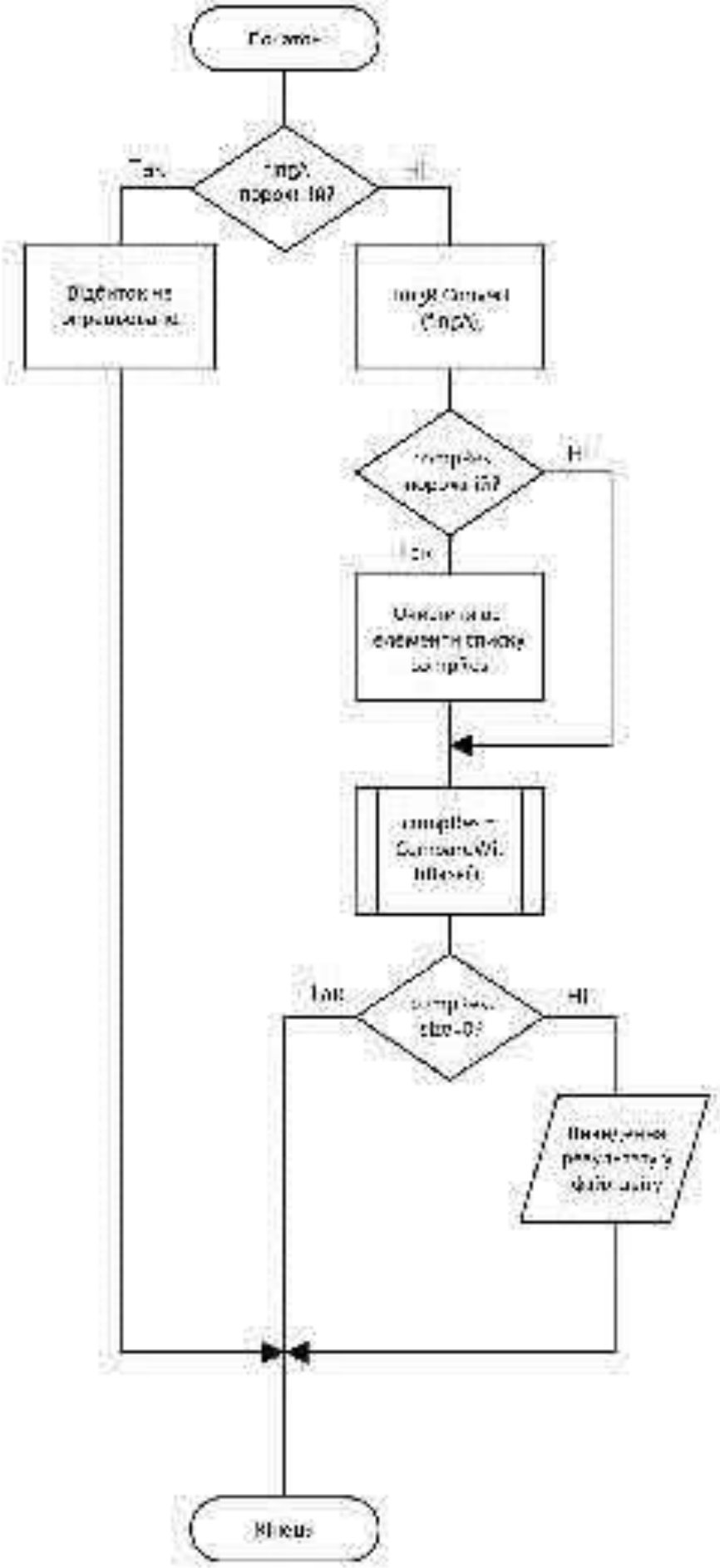

Рисунок 1.11. Блок-схема підпрограми OnBnClickedCompare

```
class TAbsFing: public list<TAbsDot>
class TAbsDot
{
public:
CPoint coord;
double alpha;
booltype;
boolshow;
};
```

де: coord – координати,

alpha – напрямок вкращі,

type – тип країни (1 – закінчення, 0 – роздвоення),

show – видимість країни (1 – видима, 0 – прихована).

Вхідні дані для даної підпрограми представлені у вигляді посилання на список точок відбитку пальця в абсолютних параметрах – TAbsFing &fng. Вихідні дані представлені у вигляді вказівника – TRelFing.

У даній підпрограмі використовуються наступні підпрограми:

1) GetAlpha – отримання напрямку з країни A в країну B [-pi pi];
2) GETS – отримання відстані між двома країнами.

Схема підпрограми «Convert» приведена на рис. 1.12.

1.11 Розробка підпрограми «CompareWithBase»

Підпрограма CompareWithBase призначена для завантаження даних з файлу бази даних, перетворення їх до відносних параметрів.

Підпрограма порівнює кожен відбиток з бази даних з відбитком, відкритим для дослідження. Результат повертається як вихідний параметр функції.

Структура TRelFing призначена для зберігання інформації про відбитки пальців у відносних параметрах. Структура TRelFing оголошена таким чином:

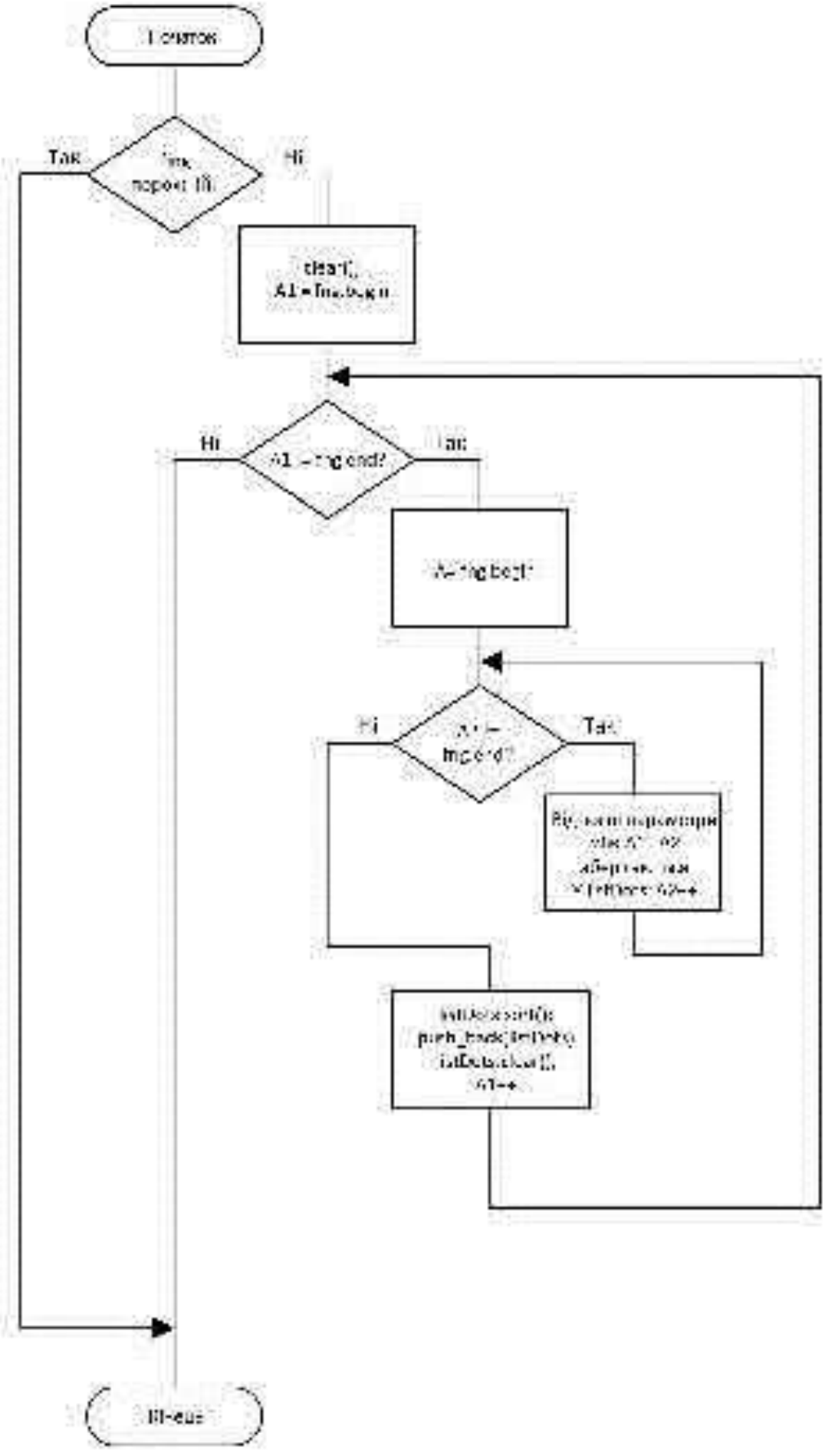

Рисунок 1.12. Блок-схема підпрограми Convert

```
class TRelFing: public list<listTRelDot>
typedef list<TRelDot> listTRelDot;
class TRelDot
{
public:
short l,a1,a2;
TAbsDot absDot;
}
```

де: l - відстань між краями,

a1 – кут між напрямком краями A і напрямком A->B,

a2 – кут між напрямком краями B і напрямком A,

absDot – абсолютні параметри (необхідно для відображення на екрані крапок, що збіглися).

Вхідні дані для даної підпрограми представлені:

TRelFing fingR – досліджуваний відбиток у відносних параметрах;

Bse – вміст бази даних з відбитками.

Вихідні дані для даної підпрограми представлені:

list<TCompareFing> – список, що містить результати порівняння відбитків для кожного відбитку з бази даних з досліджуваним відбитком.

У даній підпрограмі використовуються наступні підпрограми:

1) LoadFing – завантаження відбитку з бази даних,
2) Compare – порівняння двох відбитків.

Схема підпрограми «Compare WithBase» приведена на рис. 1.13.

1.12 Розробка підпрограми «Compare»

Підпрограма Compare призначена для порівняння двох поданих на вхід відбитків. Підпрограма реалізує задачу пошуку співпадаючих відбитків по базі даних, порівнює кожну крапку на вхідному відбитку з кожною крапкою на другому відбитку. Результат повертається як вихідний параметр функції.

Підпрограма Compare виконує:

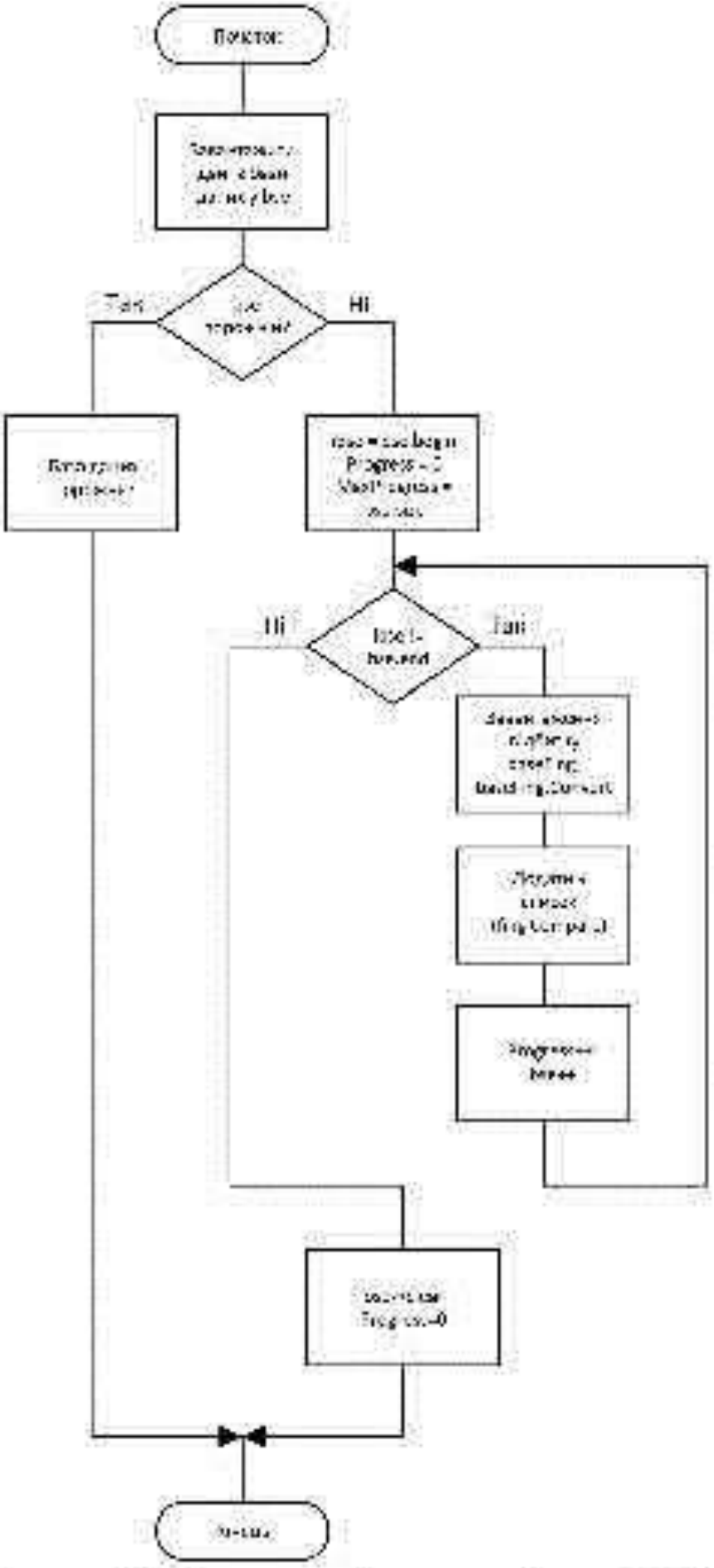

Рисунок 1.13. Блок-схема підпрограми Compare WithBase

1) порівняння двох відбитків, обчислення ступеня схожості;

2) реалізацію алгоритму розпізнавання відбитку.

Вхідні дані для даної підпрограми представлені:

TRelFing &fing – посилання на список точок відбитку пальця у відносних параметрах,

This – покажчик на список точок другого відбитку пальця у відносних параметрах.

Вихідні дані для даної підпрограми представлені:

TCompareFing – результат порівняння двох відбитків.

Схема підпрограми «Compare» приведена на рис. 1.14.

1.13 Опис контрольного прикладу

Основною метою роботи програми є пізнання особи по відбитках пальців на основі порівняння структурного представлення папілярних візерунків. Контрольний приклад повинен містити велику кількість тестових відбитків пальців, при цьому відбиток одного і того ж пальця має бути представлений як мінімум в двох екземплярах для порівняння їх між собою.

Для тесту використовувалося близько 50 відбитків різних людей і різного віку. На рис. 1.15, 1.16, 1.17 приведено декілька зображень папілярного візерунку, які передбачається порівнювати між собою і іншими відбитками в базі даних відбитків. Дані зображення отримані за допомогою чорного тонеру та аркуша білого паперу, після чого відбитки були відскановані і збережені у вигляді *.bmp файлів на комп'ютері. Отримані таким чином відбитки мають невисоку якість, тому можна повністю перевірити всі етапи роботи програми.

На рис. 1.15, 1.16 представлені відбитки одного і того ж пальця, а значить, в результаті роботи програми вони повинні збігтися. Рис. 1.17 – це відбиток іншого пальця, ніж попередні три відбитки. Підсистема розпізнавання в якості вхідних параметрів приймає результат роботи системи аналізу зображення. На рис. 1.18 показано оброблений відбиток A1, на рис. 1.19 – A2, на рис. 1.20 – B.

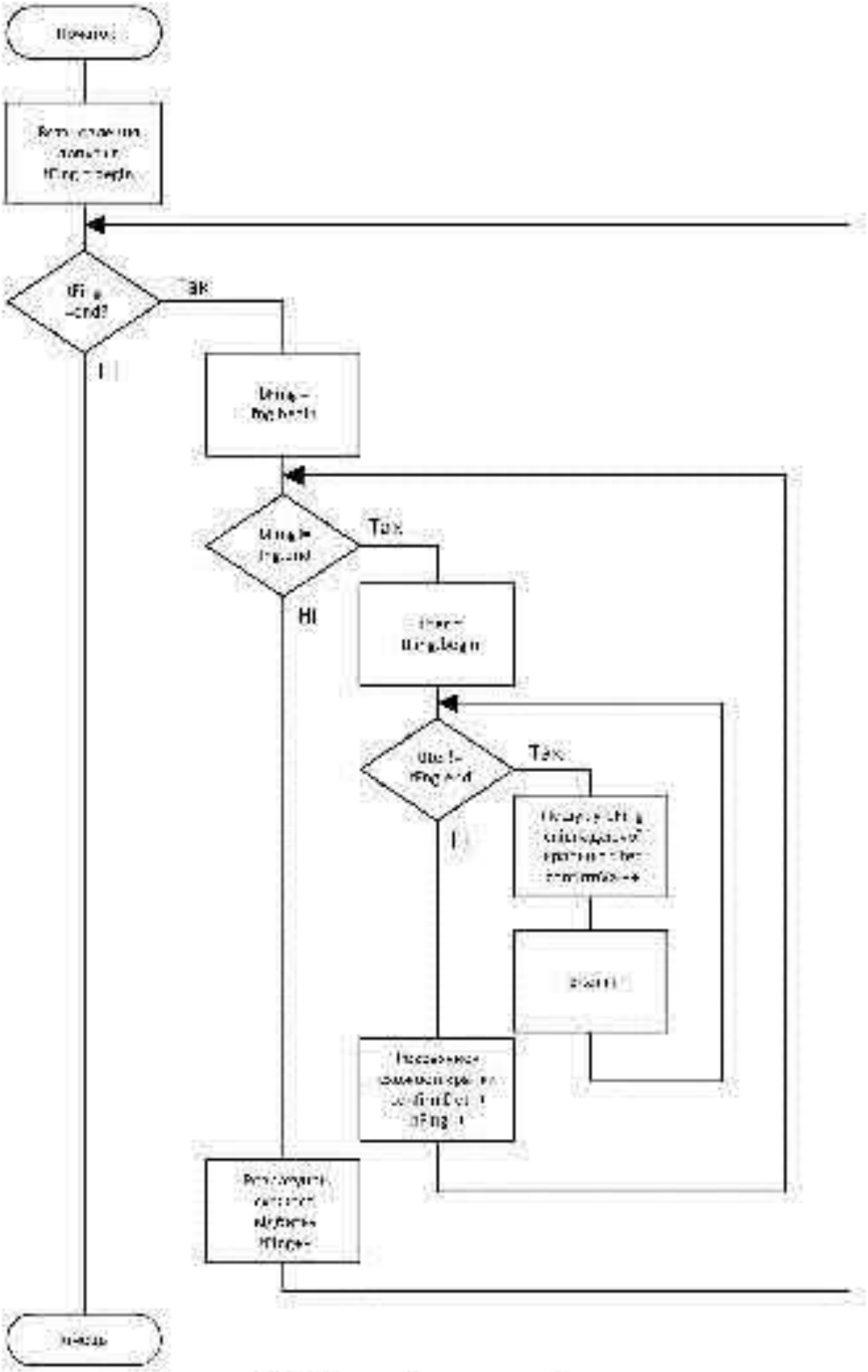

Рисунок 1.14. Схема підпрограми Compare

Рисунок 1.15. Початковий образ A1

Рисунок 1.16. Початковий образ A2

Рисунок 1.17. Початковий образ B

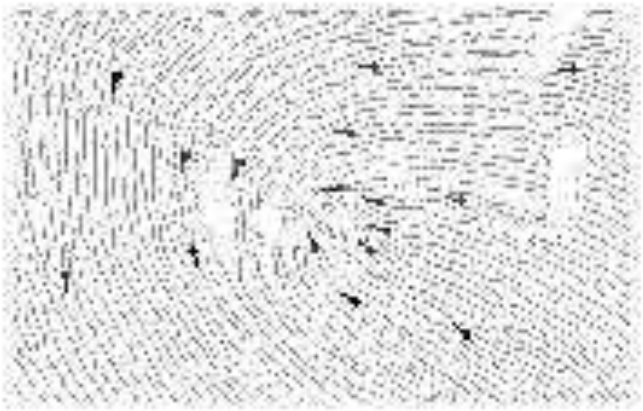

Рисунок 1.18. Оброблений образ А1

Рисунок 1.19. Оброблений образ А2

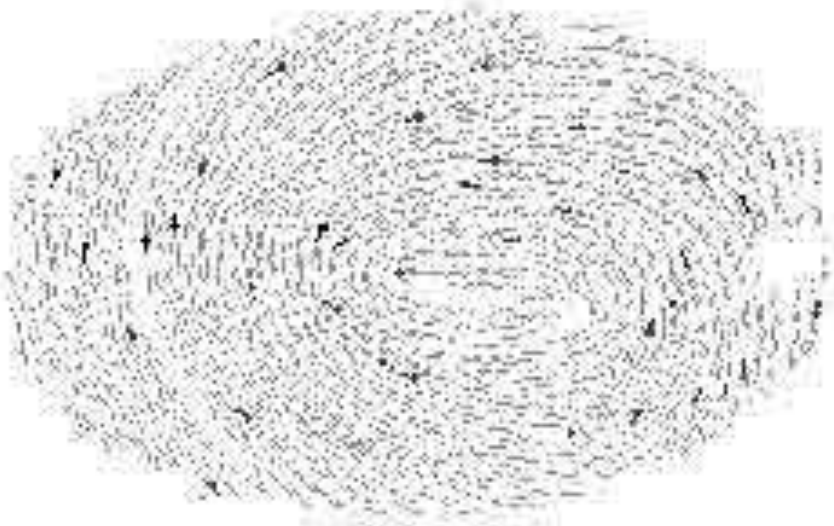

Рисунок 1.20. Оброблений образ В

При знятті відбитків для перевірки коректності роботи програми їх зображенням давалися імена, що однозначно ідентифікують палець, щоб при порівнянні по імені файлу можна було визначити чи правильно пройшло розпізнавання.

Ім'я файлу має формат:

Ім'я або номер – людина, з якої знімали відбиток;

L або R – права або ліва рука;

1,2,3,4 або 5 – палець на руці – від великого до мізинця;

символ «_»;

номер – з кожного пальця знімалося декілька відбитків.

Приклад: відбиток з ім'ям файлу 2r1_0.bmp повинен збігтися з відбитками 2r1_1.bmp і 2r1_2.bmp, але повинен відрізнятися від відбитку 2r2_0.bmp або 3r1_0.bmp.

Для випробування програмного забезпечення системи на вхід були подані описані вище тестові образи. Випробування проводилися згідно керівництву програміста і керівництву оператора, приведених в розділі 3. В результаті були отримані структурні описи представлені на рис. 1.18, 1.19, 1.20.

Отримані структурні представлення точно описують вхідні образи, що не важко перевірити візуальним порівнянням з вхідними образами. Статистично було виявлено, що на відбитках є близько 40-50 мінуцій, ця величина може змінюватися залежно від розмірів пальця. На тестових образах знайдене 19, 40, 37 відповідно.

У додатку можна побачити що відбитки A1 (1.bmp), A2 (1R1 3rotate2.bmp) схожі між собою і відбитком 1R1_1.bmp, що є вірним, оскільки всі вони є образами великого пальця правої руки однієї і тієї ж людини. Відбиток B (3l2_2.bmp) не збігається ні з одним з A1 і A2, але збігається з 3l2_1.bmp, що є також вірним результатом, це відбитки вказівного пальця правої руки іншої людини.

Тестування показало, що розроблене програмне забезпечення здатне порівнювати і відшукувати схожі відбитки, а значить є і можливість визначити

людину, якій належить, оброблювані відбиток. Отримані результати збігаються з очікуваними і збігаються з ручним порівнянням.

Окрім об'єктного опису папілярного візерунку відбитків і подальшого розпізнавання особи по відбитках пальців, реалізований в програмі алгоритм придатний для опису і розпізнавання символьної інформації і підписів. Для розпізнавання інших, відмінних від відбитків образів, в програмі досить змінити параметри виділення локальних особливостей і прибрати декілька етапів. Етап відновлення растру для символів не годиться, оскільки він спеціально розроблений для папілярного візерунку. Програма, після невеликого коректування параметрів аналізу і порівняння, була налагоджена для розпізнавання символів алфавіту.

Для перевірки був введений в базу даних набір символів змальованих на рис. 1.21, який в подальшому порівнювався з алфавітом на рис. 1.22.

Рисунок 1.21. Алфавіт для бази даних

Рисунок 1.22. Алфавіт для порівняння

1.14 Розробка керівництва програміста

1.14.1 Характеристика програмного застосунку

Програма FingerAnalyser вимагає для свого функціонування наявність в проекті файлів, що містять растрові представлення папілярного візерунку.

До складу програми входять наступні файли, необхідні для її функціонування:

1) FingerAnalyser.exe – виконуваний файл, що містить основний інтерфейс програми;
2) MFC – бібліотеки для підтримки віконного середовища;
3) blank.bmp – порожнє зображення;
4) report.txt – файл звіту, в який записуються всі результати порівняння;
5) sav/*.sav – файли із структурним представленням відбитків;
6) sav/fingbase.bse – база даних відбитків;
7) dll бібліотеки MFC.

Програма є інтерактивною, вимагаючою взаємодії з користувачем, тому час виконання окремих етапів обробки не перевищує 0.5с при використанні відповідних технічних засобів.

1.14.2 Звернення до програмного застосунку

Для запуску програми необхідно переконатися в тому, що необхідні бібліотеки MFC знаходяться в тому ж каталозі, що і виконуваний файл або в каталозі Windows/System32.

Для коректної роботи програми вона повинна знаходитися в каталозі, до якого є права на читання і запис.

Для запуску підсистеми необхідно в оболонці системи на головному вікні натиснути на кнопку «Аналіз» – для порівняння потрібного відбитку з набором наявних в базі даних, або «Запам'ятати в базу» – для внесення вказаних відбитків до бази даних.

1.14.3 Вхідні і вихідні дані для програмного застосунку

Вхідними і вихідними даними для програми є файл бази даних відбитків sav/fingbase.bse. Структура файлу:

```
src |kol| |dpi| |date| |description|
src |kol| |dpi| |date| |description|
src |kol| |dpi| |date| |description|
```

У кожному рядку файлу бази даних відбитків міститься опис одного відбитку. У таблиці 1.4 приведений формат запису у файлі бази даних.

Таблиця 1.4

Поле	Формат	Опис
src	Рядок	шлях до образу, з якого була отримана інформація
kol	Ціле	кількість крапок
dpi	Ціле	якість початкового відбитку (dot per inch)
date	Дата	дата відбитку
description	Рядок	опис

Вихідними даними для програми є файл sav/*.sav із структурним представленням, що містить статистичні характеристики мінуцій на відбитку. Цей файл має наступний формат:

x в alpha type [show]

x в alpha type [show]

x в alpha type [show]

У кожному рядку файлу структурного представлення міститься опис однієї мінуції. У таблиці 1.5 приведений формат рядка із структурним описом мінуції в абсолютних параметрах.

Таблиця 1.5

Поле	Формат	Опис
x	Ціле	Абциса мінуції на растрі
y	Ціле	Ордината мінуції на растрі
alpha	Ціле	Орієнтація мінуції на растрі
type	Байт	Тип мінуції. Роздвоєння або закінчення

Крім того, вихідною інформацією для даної підсистеми є файл звіту report.txt, що містить відбитки з бази даних, в яких була виявлена схожість з обробленим відбитком. У кожному рядку звіту міститься опис відбитку, його

ім'я, кількість крапок, що збіглися при розпізнаванні і ступінь схожості. В таблиці 1.6 приведений формат даних файлу звіту. На рис. 1.23 наведений приклад файлу звіту. Файл report.txt має наступний формат:

----- Name$_1$ -----
Count$_{i1}$ Pct$_{i1}$ Source$_{i1}$
Count$_{i2}$ Pct$_{i2}$ Source$_{i2}$
Count$_{iA}$ Pct$_{iA}$ Source$_{iA}$

Таблиця 1.6

Поле	Формат	Опис
Name	Рядок	Ім'я відбитку
Count	Ціле	Абсциса мінуції на растрі
Pct	Ціле	Ступінь схожості відбитків у відсотках, набуває значень [0, 100]
Source	Рядок	Шлях до файлу, з якого були узяті параметри
NN	Ціле	Кількість відбитків наявних в базі даних

Повідомлення, що видаються програмістові, приведені в таблиці 1.7. Фрагмент код програми на мові програмування С++ наведений у Додатку А.

Таблиця 1.7

Повідомлення	Для програміста
Відбиток не оброблений	Перш ніж запускати порівняння необхідно провести аналіз
База даних порожня	Необхідно заповнити базу даних відбитків
Відбиток не придатний для збереження в базу	На відбитку або дуже мало виявлено мінуцій (менше 10), або дуже багато (більше 80)
Жодного відбитку немає	В результаті пошуку не збігося жодного відбитку
Неможливо створити базу даних з відбитками	Можливо немає прав на запис або вільного місця на носієві
Неможливо створити файл	Можливо немає прав на запис або вільного місця на носієві
Неможливо відкрити файл	Можливо немає прав на читання або не існує запрошуваного файлу на носієві
Виявлена схожість	У базі даних були виявлені відбитки, що мають схоже представлення

Рисунок 1.23. Приклад файлу звіту

1.15 Розробка керівництва оператора застосунку

1.15.1 Призначення програмного застосунку

Програма розпізнавання користувача по відбитках пальців має ідентифікатор FingerAnalyser і призначена для автоматичної ідентифікації особи по папілярному візерунку. Програма FingerAnalyser виконує наступні функції:

1) модифікація зображення, виправлення спотворень;
2) виділення локальних особливостей – мінуцій. Формування списку мінуцій в абсолютних параметрах;
3) сортування списку абсолютних параметрів, виключення помилкових і ненадійних мінуцій;
4) конвертація абсолютних параметрів у відносні, формування списку відносних параметрів;
5) установка системи допусків для обліку кореляції зображень;
6) порівняння одного відбитку з безліччю інших.

Дана програма реалізує таке перетворення зображення, при якому дані про розташування унікальних особливостей зберігаються якнайповніше і з найменшим вмістом помилкової інформації. Створювана система полегшить розробку алгоритмів обробки зображень, спростить аналіз експериментальних даних і виявлення спільних закономірностей. Програма FingerAnalyser пред'являє наступні вимоги до програмних засобів: операційна система сімейства Windows; dll-бібліотеки MFC.

1.15.2 Виконання програмного застосунку

Інтерфейс програми представлений на рис. 1.24.

Рисунок 1.24. Інтерфейс програмного застосунку FingerAnalyser

На формі програми у візуальному вигляді представляється, після відкриття через пункт «Відкрити», початкове зображення, після натиснення на кнопку «Аналіз», скоректоване зображення і візуальне представлення структурного виду відбитку. Після цього натисненням на кнопку «Порівняти» можна відшукати в базі схожі відбитки.

При кожному аналізі відбитку створюється файл з його структурним описом. Для того, щоб помістити відбиток в базу даних відбитків, для подальшого порівняння з ним, необхідно натиснути на кнопку «Запам'ятати в базу». Для запам'ятовування в базу можна вибирати групу файлів для застосування операції запису в базу даних для усіх обраних файлів.

2 ЕКОНОМІЧНА ЧАСТИНА

2.1 Резюме

В даному дипломному проекті розроблені програмні засоби для порівняння папілярних візерунків відбитків пальців

Ефективність кожного програмного продукту визначається його якістю та ефективністю процесу розробки. Якість програмного продукту визначається наступними складовими: з точки зору користувача; з позиції використання ресурсів; виконання вимог до програмного забезпечення.

Оцінка якості програмного продукту з точки зору користувача визначається необхідним на стадії функціонування розміром оперативної пам'яті комп'ютера, витратами машинного часу, пропускною спроможністю каналів передачі даних. Оцінка якості програмного продукту включає визначення трудомісткості і вартості його створення.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога.

Таблиця 2.1. Каталог аналогів

Найменування ПП	Обсяг функцій ПП – V_a, ум. машинних команд
1. ПП автоматизованих розрахунків	1300 – 8600
2. ПП загальної математики і ПП імітаційного моделювання	1800 – 8800
3. ПП організації обчислювального процесу	13000 – 10200

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт (для обраного варіанта виділено сірим кольором).

Після вибору аналогу програмного продукту, що містить Vo в умовних машинних командах, трудомісткість визначається на основі табл. 2.2

Таблиця 2.2. Трудомісткість виконання програмного продукту

Обсяг ПП, тис. умов. машинних команд	Норма часу, люд./год.
100	229
200	244
300	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 229 х 0,8 = 183 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{пр} = T^{*}_{пр} \times L_i \times K_н; \quad T_{пр} = T^{*}_{пр} \times L_i \times K_н; \quad T_{пр} = T^{*}_{пр} \times L_i \times K_н \times K_т; \qquad (2.1)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.2); Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.3); Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.4).

Таблиця 2.2. Значення питомих коефіцієнтів трудомісткості стадій в загальній трудомісткості розробки ПП

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для даного варіанта виділено сірим кольором.

Таблиця 2.3. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $K_н$
А	Принципово нове ПЗ	1,75 – 1,2
Б	Розвиток визначеного параметричного ряду ПЗ	1,0 – 0,8
В	ПЗ, що має аналог	0,7

Таблиця 2.4. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розробленого ПЗ типовими програмами, %	Значення $K_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для даного варіанта виділено сірим кольором.

Тепер розрахуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$T_{тз} = Tар * L_1 * K_н = 183 * 0,12 * 0,7 = 15,4 \quad (\text{людгодин}) \qquad (2.2)$$

Трудомісткість розробки технічного проекту

$$T_{тп} = Tар * L_2 * K_н = 183 * 0,11 * 0,7 = 14,9 \quad (\text{людгодин}) \qquad (2.3)$$

Трудомісткість розробки робочого проекту

$$T_{рп} = Tар * L_3 * K_н * K_т = 183 * 0,61 * 0,7 * 0,7 = 54,7 \quad (\text{людгодин}) \qquad (2.4)$$

Для подальших розрахунків визначається кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$= 2 (стр.), розробка ТП $N_{тп}$=20 (стр.), розробка робочого проекту $N_{рп}$=18 (стр.), пояснювальна записка відповідно $N_{пз}$= 49 (стр.) Розрахунок зведений у таблицю 2.5.

Таблиця 2.5. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин		
1	2	3	4
1. ТЗ	$T_{рз}$=1,5*2=3	$T_{кз}$=0,7·$N_{тз}$= 0,7*2=1,4	$T_{сз}$=0,15·$N_{тз}$= 0,15*2=0,3
2. Розробка ТП	$T_{рто}$=1,5*20=30	$T_{кз}$=0,7·$N_{тто}$= 0,7*20=14	$T_{сз}$=0,15·$N_{тто}$= 0,15*20=3
3. Розробка РП	$T_{рро}$=1,5*18=27	$T_{кз}$=0,7·$N_{рро}$= 0,7*18=12,6	$T_{сз}$=0,15·$N_{рро}$= 0,15*18=2,7
4. Розробка ПЗ	$T_{пз}$=1,5·$N_{пз}$= 1,5*49=73,5	$T_{кз}$=0,7·$N_{тз}$= 0,7*49=34,3	$T_{сз}$=0,15·$N_{пз}$= 0,15*49=7,35
Усього, в т.ч.:	133,5	62,3	13,35
- на розробку	$\Sigma T_р$=133,5		
- контроль керівника		$\Sigma T_{кз}$=62,3	
- нормоконтроль			$\Sigma T_{сз}$=13,35

2.3 Розрахунок ціни програмного продукту

У цьому розділі для визначення ціни розраховано основну заробітну плату виконавців, матеріальні витрати, вартість машино-години і витрати на розробку.

Таблиця 2.6. Розрахунок основної заробітної плати виконавців

Найменування робіт	Трудомісткість робіт, годин	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1. Розробка ПП	133,5	39.26	5241
2. Контроль керівника	62,3	50.00	3115
3. Нормоконтроль	13,35	50.00	667
Усього	-	-	ΣЗо= 9023

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведено у таблицю 2.7.

Таблиця 2.7. Розрахунок матеріальних витрат на розробку ПЗ

Найменування матер. витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	89	0,50	44,5
Разом	-	-	-	$В_{м1}$=44,5
Транспортно-заготівельні витрати (10%)				$Втр\_з = 0,1 \times Вм1$ = 4,45
Усього				$В_м$=$В_{м1}$+$В_{тр\_з}$= 44,5+4,45= 48,95

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості цілому програмного продукту за формою.

Таблиця 2.8. Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	44,5	$В_м$ (див. табл. 2.7)
2. Основна заробітна плата	9023	$З_о$ (див. табл. 2.6)
3. Додаткова заробітна плата	1353	$Зд = 0,15 \times Зо = 0,15 * 9023$
4 Відрахування до єдиного фонду соціального внеску	1282	$Вес.в. = 0,22 \times (Зо + Зд) = 0,22 * (9023 + 1353)$
5. Накладні витрати	2706	$Внак. = 0,3 \times Зо = 0,3 * 9023$
6. Повна собівартість	14408	$С_{пов} = В_м + З_о + З_д + В_{ссв} + В_{нак} = 44,5 + 9023 + 1353 + 1282 + 2706$

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$$П = (С_п * Р)/100 = (14408 * 13)/100 = 1873 \text{ грн.} \quad (2.5)$$

де р – плановий рівень рентабельності (10-15%).

Оптова ціна (споживча вартість) визначається по формулі:

$$Ц_п = С_п + П = 14408 + 1873 = 16281 \text{ грн.} \quad (2.6)$$

Податок на додану вартість визначаємо по наступній формулі:

$$ПДВ = 0,2 * Ц_п = 0,2 * 16281 = 3256 \text{ грн.} \quad (2.7)$$

Виходячи з отриманих даних, ціна реалізації розробленого програмного продукту на основі наступної формули, становитиме:

$$Ц_р = Ц_п + ПДВ = 16281 * 3256 = 19537 \text{ грн.} \quad (2.8)$$

3 ОХОРОНА ПРАЦІ

Питання охорони праці людини необхідно вирішувати на всіх стадіях трудового процесу незалежно від виду професійної діяльності Забезпечення безпечних і здорових умов праці в значній мірі залежить від правильної оцінки небезпечних,шкідливих виробничих факторів.

Однакові по складності зміни в організмі людини можуть бути викликані різними причинами. Це можуть бути фактори виробничого середовища, надмірне фізичне і розумове навантаження, нервово-емоційна напруга, а також різне сполучення цих причин.

При виконанні робіт на робочому місці програміста-розробника потрібно дотримуватись встановлених вимог безпеки праці, а саме нормативно-правового акту з охорони праці (НПАОП 0.00-7.15-18) «Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», а також НПАОП 0.00-1.28-10 "Правила охорони праці під час експлуатації електронно-обчислювальних машин", затверджені наказом Державного комітету України з промислової безпеки, охорони праці та гірничого нагляду від 26.03.2010 № 65.

У даному розділі ДП вирішується питання охорони праці програміста на стадії розробки ним програмних засобів для папілярних візерунків відбитків пальців.

3.1 Аналіз та безпека умов роботи працівника на робочому місці

Аналіз умов праці показує, що у приміщенні лабораторії на програміста можуть негативно впливати наступні фізичні та психофізіологічні фактори: підвищена або знижена температура повітря робочої зони; підвищена або знижена вологість повітря; недостатня освітленість робочого місця; підвищений рівень шуму на робочому місці; підвищена іонізація повітря; підвищений рівень електромагнітних випромінювань; нервово-психічні перевантаження (розумова перенапруга, перенапруга аналізаторів); фізичні перевантаження (одноманітна поза викликає статичну втому). Рекомендації, щодо усунення шкідливих

факторів або максимальне зниження їх впливу на працівника наведеного у роботі:

– Для зменшення втоми очей рекомендується використання правильного освітлення – гарне природнє освітлення, у темний час доби лампа повинна освітлювати тільки кімнату, але не екран монітора – це допоможе уникнути відблисків, що ускладнюють роботу

– Для того, щоб зменшити шкідливий вплив незручної пози та довгого знаходження у сидячому положенні, необхідно правильно підібрати робочі меблі. Вона допоможе зберегти правильне положення тіла під час роботи за комп'ютером і зменшити навантаження на м'язи.

– До рекомендацій стосовно зменшення навантаження на нервову систему відносяться прогулянку по свіжому повітрі, спілкування з людьми у реальному світі та також потрібно робити перерви при роботі з комп'ютерною технікою.

3.1.1 Вимоги до приміщення

Приміщення, в якому працює програміст, має загальну площу 20 м^2, висоту стелі 3 м. У приміщенні знаходиться 7 робочих місць з ПК. Слід відзначити, що площа одного робочого місця оператора ПК не повинна бути меншою за 6м^2, а об'єм не меньший за 20м^3, тобто площі та об'єму даного приміщення не вистачає для розташування 7 робочих місць операторів ПК.

3.1.2 Мікроклімат робочої зони програміста

Робота програміста за енергозатратами відноситься до категорії легких робіт Іа, Іб, тому повинні дотримуватися наступні вимоги ДСН 3.3.6.042-99:

- оптимальна температура повітря – 22° С (допустима – 20-24° С);
- оптимальна відносна вологість – 40-60% (допустима –не більш 75%);
- швидкість руху повітряне більш 0,1 м/с.

Ці вимоги виконуються у даному проекті.

3.1.3 Освітлення робочого місця

В приміщенні, де використовується праця на комп'ютері, наявні елементи природного та штучного освітлення.

Нормованим параметром природного освітлення згідно ДБН В.2.5–28 – 2006 є коефіцієнт природного освітлення (КПО). Для освітлення приміщення, у якому працює програміст, використовується змішане освітлення, тобто сполучення природного й штучного освітлення. Для загального освітлення приміщення, де перебуває робоче місце програміста, використовуються газорозрядні лампи типу ЛД. Нормами для даних робіт установлена необхідна освітленість робочого місця ЕН=300 лк (для робіт високої точності, коли найменший розмір об'єкта розрізнення дорівнює 0,3 – 0,5мм).

3.1.4 Вплив шуму на програміста

Як було вказано вище, в лабораторії знаходиться сім робочих місць з ПК, кожне з яких устатковане монітором, вінчестером в системному блоці, трьома вентиляторами системи охолоджування ПК та клавіатурою. Крім того поряд працює периферійна техніка. Таким чином у приміщенні мають місце шуми механічного і аеродинамічного походження, широкосмугові із аперіодичним підсиленням при роботі принтерів.

3.1.5 Виробничі випромінювання

Допустимі значення параметрів неіонізуючих електромагнітних випромінювань від монітору комп'ютера представлені в таблиці 1.

Таблиця 3.1. Допустимі значення неіонізуючих електромагнітних випромінювань

Найменування параметра	Допустимі значення
Напруженість електричної складової електромагнітного поля на відстані 50 см від поверхні відео-монітора	10 В/м
Напруженість магнітної складової електромагнітного поля на відстані 50 см від поверхні відео-монітора	0,3 А/м
Напруженість електростатичного поля не повинна перевищувати	20кВ/м

На відстані 5-10 см від екрана і корпусу монітора рівні напруженості можуть досягати 140 В/м по електричній складовій, що значно перевищує допустимі значення.

3.1.6 Електробезпека. Статична електрика

Приміщення лабораторії за небезпекою ураження електричним струмом можна віднести до 1 класу, тобто це приміщення без підвищеної небезпеки (сухе, без пилу, з нормальною температурою повітря, ізолюваними підлогами і малим числом заземлених приладів). На протязі роботи на корпусі комп'ютера накопичується статична електрика. На відстані 5-10 см від екрана напруженість електростатичного поля складає 60-280 кВ/м, тобто в 10 разів перевищує норму 20 кВ/м.

3.1.7 Організація робочого місця

Кожне робоче місце обладнане робочім столом площею 1,2 м^2, стільцем та персональним комп'ютером, що складається з монітора, системного блоку, клавіатури та миші. Обладнання і організація робочого місця з ПК мають забезпечувати відповідність конструкцій всіх елементів робочого місця та їх взаємного розташування ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності (ГОСТ 12.2.032-78, ГОСТ 22269-76, ГОСТ 21889-76). Дисплей розташований так, що його верхній край перебуває на рівні очей на відстані близько 70 см, що укладається в у припустимі рамки від 60 до 90 см. Частота мерехтіння екрана fмер=100 Гц, що відповідає умові fмер ≥70 Гц.

Робоче місце розташоване перпендикулярно віконним прорізам, це зроблено з тією метою, щоб виключити пряму й відбиту мерехливість екрана від вікон і приладів штучного освітлення, якими є лампи накалювання. Обладнання і організація робочого місця з ВДТ мають забезпечувати відповідність конструкцій всіх елементів робочого місця та їх взаємного розташування, ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності (ГОСТ 12.2.032.-78, ГОСТ 22269.-76, ГОСТ 21889-76).

3.2 Пожежна безпека

Ступінь вогнестійкості будинків приймається в залежності від їхнього призначення, категорії по вибухопожежній і пожежній небезпеці, по поверховості, площі поверху в межах пожежного відсіку згідно НАПБ Б.03.002-

2007. Будинок, у якому знаходиться лабораторія по пожежній небезпеці будівельних конструкцій відноситься до категорії К1 (малопожежонебезпечні), оскільки тут присутні займисті (книги, документи, меблі, оргтехніка і т.д.) і тяжкогорючі речовини (сейфи, різне устаткування і т.д.), що при взаємодії з вогнем можуть горіти без вибуху.

Причинами виникнення пожежі можуть бути: несправності електропроводки, розеток і вимикачів які можуть привести до короткого замикання або пробою ізоляції; використання ушкоджених (несправних) електроприладів; використання в приміщенні електронагрівальних приладів з відкритими нагрівальними елементами; виникнення пожежі внаслідок влучення блискавки в будинок; загорання будинку внаслідок зовнішніх впливів; неакуратне поводження з вогнем і недотриманням мір пожежної безпеки.

Для гасіння пожеж на робочому місці використовують вуглекислотні та порошкові вогнегасники.

– Вуглекислотні вогнегасники випускаються якручні (ВВК-5).

– Порошкові вогнегасники ВП-2, ВП-5, ВП-10 та інші.

Наявність первинних засобів пожежогасіння і вогнегасників, їхня кількість і зміст відповідає вимогам ГОСТ 12.4.009-75 і ISO3941-77. У приміщенні виконуються усі вимоги по пожежній безпеці відповідно до вимог НАПБ А.0.001-95 "Правила пожежної безпеки в Україні". У приміщенні також має ться план евакуації на випадок виникнення пожежі. Час евакуації відповідає вимогі СНиП 2.01.02-85, а максимальне видалення робочих місць від евакуаційних виходів відповідає СНиП 2.09.02-85.

ВИСНОВКИ

В результаті виконаної дипломної роботи реалізовано метод, який може бути застосовано для автоматизації пошуку схожих відбитків пальців і створено програмне забезпечення для реалізації даного методу. Створений програмний застосунок дозволяє за прийнятний час виділяти локальні особливості відбитків пальця (папілярний візерунок). В порівнянні з ручним визначенням «на око» по ключових ділянках початкового зображення отримано значний виграш в швидкості і зручності використання. Отримувані статистичні характеристики досить повно описують зображення і дозволяють провести розпізнавання з високим ступенем точності.

Робота включала в себе розробку і реалізацію алгоритму формування відносних параметрів для мінюцій відбитків пальців, підбір параметрів системи допусків і критеріїв подібності при порівнянні двох відбитків, розробку та реалізацію алгоритму пошуку схожих відбитків на основі відносних параметрів мінюцій і розроблених критеріїв подібності.

Розроблена підсистема є некід'ємною частиною системи ідентифікації особи, призначеної для виявлення схожості між двома зображеннями відбитку пальця. В результаті розпізнавання можна встановити особу людини, що привласнила палець, що може використовуватися при вході в систему. За допомогою підсистеми розпізнавання можна значно знизити рівень впливу зсуву і перенесення відбитку пальців, а також шумів і спотворень в зображенні.

Створене програмне забезпечення слід розглядати як тестову систему, призначену для виявлення емпіричних закономірностей в наочній області і подальшої розробки у напрямі автоматизації процесу ідентифікації особи користувача за папілярним візерунком відбитків пальців.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Загородний В.И. Комплексная защита информации в компьютерных системах // Учебное пособие. – М.: Логос, 2001, – 264 с.
2. Кухарев Г.А. Биометрические системы: Методы и средства идентификации личности человека. – СПб.: Политехника, 2001. – 240 с.
3. Гудков В.Ю. Метод параллельных цепей для распознавания изображений отпечатков пальцев // Системы и средства информатики. – 2013. – С. 35-49.
4. Самищенко С.С. Атлас необычных папиллярных узоров. - М.: Юриспруденция, 2001. – 307 с.
5. Насонов А.В., Крылов А.С., Ушмаев О.С. Применение метода суперразрешения для биометрических задач распознавания лиц в видеопотоке // Системы высокой доступности. – 2009. – Т. 1. – С. 26-34.
6. Ушмаев О.С. Адаптация биометрической системы к искажающим факторам на примере дактилоскопической идентификации // Информатика и её применения. – 2009. – Т. 3, №. 2. – С. 25-33.
7. Клаковіч Л. М., С. М. Левицька Теорія алгоритмів: Навчальний посібник. – Друге видання, доповнене / Л. М. Клаковіч, С. М. Левицька. – Львів: Видавничий центр ЛНУ ім. Івана Франка, 2015 – 161 с.
8. Зацерковний В. І. Алгоритмізація та програмування: навчальний посібник / В. І. Зацерковний, В. І. Гур'єв, І. В. Фірсова. – Ніжин: НДУ ім. М. Гоголя, 2013 – 302 с.
9. Ю.О. Гришко, О.С. Шкільняк. Алгоритми обчислювальної геометрії. Навчальний посібник. – 2020 – 153 с.
10. Дьюхарст Программирование на C++ / Дьюхарст, Старк Стефан; Кэти. – М.: ДиаСофт, 2015. – 272 с.
11. Фридман, А. C/C++. Архив программ / А. Фридман, Л. Кландер, М. Михаэлис, и др. – М.: ЗАО Издательство БИНОМ, 2016. – 640 с.
12. Брюс Эккель. Философия C++. Введение в стандартный C++ – СПб.: Питер, 2004. – 577 с.

ДОДАТОК А. Фрагмент тексту модуля TAnalysePicture.cpp

```
#include "StdAfx.h"
#include "TAnalysePicture.h"
TAnalysePicture :: TAnalysePicture (const CString src, CDC * screen)
{
pic = new TFlingPicture (screen);
err = -1;
if(!pic-> Load (src)) err = 0;
pic-> Rectangle (CPoint (0, 0), pic-> GetSize (), 10);
srcImg = src;
impPic = new TFlingPicture (screen);
impPic-> Load (src);
pic2 = new TFlingPicture (screen);
pic2-> Load (BLANK);
}
TAnalysePicture :: ~ TAnalysePicture (void)
{
delete (impPic);
delete (pic2);
delete (pic);
}
// Код помилки
int TAnalysePicture :: getErr ()
{
return err;
}
// Повідомлення помилки
CString TAnalysePicture :: getErrMsg ()
{
CString msg = "";
switch (err)
{
case -1: {msg = "Помилок при завантаженні зображення немає"; break;}
case 0: {msg = "Зображення не завантажене"; break;}
case 1: {msg = "Виникла помилка при завантаженні зображення"; break;}
default: {msg = "Невідома помилка";}
}
return msg;
}
// Обробка завантаженого зображення і отримання образу
TAbsFling TAnalysePicture :: AnalysePicture ()
{
TAbsFling rel, rel2;
if(err != -1)
{
if(MESSAGEBOUT) MessageBox (NULL, getErrMsg (), "Помилка", MB_OK);
return rel;
}
int preuCol;
int changeN = 0; // Лічильник проведених змін на зображенні
list < TMapBDob> map ; // Карта точок, що належать лініям
list < TMapBDob> :: Iterator imap; // Iterator для map
map = LookPic (); // сканування картинки і знаходження ліній на ній
do {
changeN = 0;
preuCol = (int) map.size ();
imap = map.begin ();
do // Зображення можна модифікувати
if(imap-> pr1) // Лінія потребує обробки
changeN += ChangeLine (imap, map); // Обробка (перетворення) зображення
imap ++; // Перехід для обробки наступної лінії
} While (imap != Map.end ()); // Зображення можна модифікувати
} While (preuCol < 0.1 * map.size ()); // Зображення можна модифікувати
map = LookPic (); // сканування картинки і знаходження ліній на ній
imap = map.begin ();
```

```
do // Зображення можна модифікувати
rel.merge (ReadPic(imap));
imap ++; // Перехід для обробки наступної лінії
} while (imap != Map.end ()); // Зображення можна модифікувати
//////////////// Фільтрування отриманих точок ////////////////
// від'єднаються близькі спрямовані в протилежні рядки //
/////// а також точки ліворуч і праворуч від лінії немає ліній ///////
int lenDots = 0; // число відібраних точок
lenD ots = DotsFilter (rel); // Фільтрування отриманих точок
rel2.clear ();
for (TAbsRing ::Iterator iter = rel.begin (); iter != rel.end (); iter ++)
{
if(!iter-> show) continue;
// малювання знайдених точок (колір залежить від розділення рівня)
COLORREF col = (iter-> type)? 0xFF0000 : 0x000000;
pic2-> Line (iter-> coord, iter-> coord, 5, col);
pic2-> Line (iter-> coord,
CPoint (iter-> coord.x + (int) (10.0 * cos (iter-> alpha)), iter-> coord.y - (int) (10.0 * sin (iter-> alpha))),
2, col);
rel2.push_back (* iter);
}
rel.clear ();
return rel2;
}
TAbsRing TAnalysePicture :: ReadPic (list<TMap BDot> ::Iterator _dot)
// Знаходження на зображенні спец. точок
{
TAbsRing relRing; // Образ відбитка в абсолютних координатах
int kol = 0; // кількість пройдених точок
int vec = 0; // напрямок пошуку чергової точки
int lenS = 0; // Поточна кількість коротких векторів
CPoint A, // Початок вектора
B; // Кінець вектора
TAbsRing vecDotS; // масив точок для коротких векторів
TAbsRing vecDotL; // масив точок для довгих векторів
TAbsRing historyDotL; // історія точок для довгих векторів
TAbsDot _tmpDotRing, bestDot;
TAbsRing ::Iterator iter;
double alpha; // напрямок вектора (в радіанах)
int stopKol = 2000; // межа кроків
int rel = 0; // лічильник кроків після проходження початкової точки
bool homeOuter = false; // ознака закінчення обробки
A = _dot-> coord; B = _dot-> coord;
CPoint olddot, dot = _dot-> coord; // Поточна точка на лінії
do {
// основний цикл обробки,
// варіанти завершення циклу
// триває до тих пір , поки вся лінія не буде пройдена (нормальний варіант)
// зациклення (ненормальний варіант, % кількох)
olddot = dot;
dot = pic-> NextDot(0) (dot, vec); // Пошук наступної точки за годинниковою стрілкою
if(dot.x == olddot.x && dot.y == olddot.y)
{// становище точки не змінилося => вихід //
CString s;
s.Format ("x = %d, y = %d, kol =%d", dot.x, dot.y, kol);
if(MESSAGBOUT) MessageBox (0, "положення точки не змінилося => вихід \n" + s, "", MB_OK);
return relRing;
}
kol ++; // підрахунок пройдених точок
if(kol % LEN_S == 0)
{// з'явився новий короткий вектор
lenS ++;
A = B;
B = dot;
pic2-> Line (A, B, 1, 0x999999);
_tmpDotRing.coord = A;
alpha = GetAlpha (A, B);
// розрахунок локального напрямку між KOL_S пікселями (напрямок короткого вектора)
```

```
double dAlpha = 0.0; // Різниця кутів
if(vecDoIS.size () > 0) // в списку можна взяти попередні значення
dAlpha = alpha - vecDoIS.begin () -> alpha;
/ ** / if(abs (dAlpha)> = M_PI) // різниця між новим кутом і попереднім ненормальна!
{// необхідно скорегувати поточну alpha
/ ** / if(dAlpha <0.0)
{
while (abs (dAlpha)> M_PI)
{
alpha + = 2.0 * M_PI;
dAlpha + = 2.0 * M_PI;
}
} else
{
while (dAlpha> = M_PI)
{
alpha - = 2.0 * M_PI;
dAlpha - = 2.0 * M_PI;
}
}
}
_tmpDoIFing.alpha = alpha; // запам'ятовування напрямку з точки A //
vecDoIS.push_front (_tmpDoIFing);
//////// перевіряємо два сусідніх досвіх вектора з умовою що ////////
//////// пройдено досить точок, щоб порівнювати досві вектора ////
if(vecDoIS.size () < K0 L_S) continue;
// Обчислення середнього напрямку LEN_L коротких векторів //
// запис даних по досвому вектору ////////////// ///////
double sumAlpha = 0.0;
iter = vecDoIS.begin ();
vecDoIL.clear (); // перераховуємо досві вектора
for (int i = 0; i < K0 L_S; i ++)
{
sumAlpha + = iter-> alpha;
if((i + 1)% LEN_L == 0)
{
_tmpDoIFing = * iter;
_tmpDoIFing.alpha = sumAlpha / LEN_L;
vecDoIL.push_back (_tmpDoIFing);
sumAlpha = 0.0;
}
iter ++;
}
if(abs (vecDoIL.begin () -> alpha)> 3 * 2 * M_PI)
{// занадто багато оборотів //
CString s;
s.Format ("alpha =% .2f", vecDoIL.begin () -> alpha * 180);
if(MESSAGE BO UT) MessageBox (0, "занадто багато оборотів \ n" + s, "", MB_0 K);
return retFing;
}
// перевіряємо два сусідніх досвіх вектора //
dAlpha = vecDoIL.begin () -> alpha - (++ vecDoIL.begin ()) -> alpha;
if(abs (dAlpha)> (TEST_ALPHA / 180.0 * M_PI)) // сильний вимін //
{
if(historyDoIL.empty ())
{// збереження стану //
bestDoI.alpha = 0.0;
}
if(dAlpha> 0) // розділення
alpha = (vecDoIL.begin () -> alpha - M_PI + (++ vecDoIL.begin ()) -> alpha) / 2.0;
else // злиття
alpha = (vecDoIL.begin () -> alpha + M_PI + (++ vecDoIL.begin ()) -> alpha) / 2.0;
_tmpDoIFing = vecDoIL.front ();
_tmpDoIFing.alpha = alpha; // напрямок СТ (специфічна точка) //
_tmpDoIFing.type = dAlpha <0; // тип СТ //
historyDoIL.push_front (_tmpDoIFing);
if(bestDoI.alpha <= abs (dAlpha))
{
```

```
besIDot.coord = _impDotFing.coord;
besIDot.alpha = abs(dAlpha);
}
}
else // сильний вилин //
{
if(!historyDotL.empty()) // було пройдено сильний вилин
{
alpha = 0;
for(iter = historyDotL.begin(); iter != historyDotL.end(); iter++)
alpha += iter->alpha;
alpha /= historyDotL.size(); // середнє значення в пройденій СТ
iter = historyDotL.begin();
for(unsigned int i = 0; i < (historyDotL.size() / 2); i++) iter++;
// CPoint vdot = iter->coord; // найбільш ймовірна точка для СТ
CPoint vdot = besIDot.coord; // найбільш ймовірна точка для СТ
// Якщо розкоментувати ці рядки, то виключені точки мають продовження
// CPoint dotForAccept = FindAcceptDot(vdot, alpha, iter->type);
// if(dotForAccept.x == -1)
{// точка не має продовження, запам'ятовуємо ///
_impDotFing.alpha = ChangeAlphaInterval(alpha);
_impDotFing.coord = vdot;
_impDotFing.show = true;
_impDotFing.type = historyDotL.begin()->type;
relFing.push_back(_impDotFing);
}
historyDotL.clear();
slopKol += (kol * 1.5 > slopKol)? 1000: 0;
}
}
}
if(dot.x == _dot->coord.x && dot.y == _dot->coord.y)
{// ймовірно обхід лінії завершено
if(kol <= 2)
{// лінія підозріло коротка
CString s;
s.Format("%d", kol);
if(MESSAGEBOUT)MessageBox(0, "kol <= 2 kol =" + s, "", MB_OK);
return relFing;
} else
{
homeOuter = true; // пройти необхідно далі почему
slopKol = kol + K0L_L * LEN_L * LEN_S;
}
}
if(homeOuter) rel++;
} while (rel<(LEN_L * LEN_S * K0L_L) && rel<slopKol && kol <= slopKol);
_dot->pr1 = false;
_dot->pr2 = false;
return relFing;
}
list<TMapBDot> TAnalysePicture :: LookPic()
// Поліксельно "пробігання" по картинці!
// запам'ятовування чорних крапок, після знаходження чорної точки
// заливка всієї лінії в колір фону (видалення лінії з картинки)
{
list<TMapBDot> map;
TMapBDot dot;
impPic->Copy(*pic);
for(int i = 0; i<pic->GetSize().y; i++)
for(int i = 0; i<pic->GetSize().x; i++)
{
if(!impPic->GetPixel(j,i)) // знайдено чорний піксель
{
dot.coord.x = i; dot.coord.y = i;
dot.pr1 = dot.pr2 = true;
map.push_back(dot);
impPic->FloodFill(j,i,0xffffff); // видалення лінії
```

```
}
}
ImpPic-> Copy (*pic);
return map;
}
int TAnalysePicture :: ChangeLine (list<TMapElDot> ::Iterator _dot, list<TMapElDot> &_map)
// Обробка картинки, // зміна
// Обробка лінії, на яку вказує imap
// Виправлення псевдо-роздвоєнь і псевдо-закінчень на вказаній лінії
{
int change N = 0; // кількість модифікацій на лінії
int kol = 0; // кількість пройдених точок
int uec = 0; // напрямок пошуку чергової точки
int lekS = 0; // Поточна кількість коротких векторів
C Point A, // Початок вектора
B; // Кінець вектора
TAbs Ring uecDotS; // масив точок для коротких векторів
TAbs Ring uecDotL; // масив точок для довгих векторів
TAbs Ring history DotL; // історія точок для довгих векторів
TAbs Dot _ImpDotRing ;
TAbs Ring ::Iterator iter;
TAbs Dot reselDot, besID ot;
double alpha; // напрямок вектора (в радіанах)
int stopKol = 1500; // межа кроків
int ret = 0; // лічильник кроків після проходження початкової точки
bool homeOuter = false; // ознака закінчення обробки
_dot-> pr1 = false;
A = _dot-> coord; B = _dot-> coord;
C Point olddot, dot = _dot-> coord; // Поточна точка на лінії
do {
// основний цикл обробки,
// варіанти завершення циклу
// триває до тих пір поки вся лінія не буде пройдена (нормальний варіант)
// зациклення (ненормальний варіант, % кількість)
//
olddot = dot;
dot = pic-> NextDotCW (dot, uec); // Пошук наступної точки за годинниковою стрілкою
if(dot.x == olddot.x && dot.y == olddot.y)
{// становище точки не змінилося => вихід //
CString s;
s.Format ("x = %d, y = %d, kol = %d", dot.x, dot.y, kol);
if(MESSAGE BOUT) MessageBox (0, "положення точки не змінилося => вихід \n" + s, "", MB_O K);
return changeN;
}
kol ++; // підрахунок пройдених точок
if(kol% LEN_S == 0)
{// з'явився новий короткий вектор
lekS ++;
A = B;
B = dot;
// pic2-> Line (A, B, 1, 0x999999);
_ImpDotRing.coord = A;
alpha = GetAlpha (A, B);
// розрахунок локального напрямку між KO L_S пікселями (напрямок короткого вектора)
double dAlpha = 0.0; // Різниця кутів
if(uecDotS.size ()> 0) // в списку можна взяти попереднє значення
dAlpha = alpha - uecDotS.begin () -> alpha;
/ ** / if(abs (dAlpha)> M_PI) // різниця між новим кутом і попереднім ненормальна!
{// необхідно скоригувати поточну alpha
/ ** / if(dAlpha <0.0)
{
while (abs (dAlpha)> M_PI)
{
alpha + = 2.0 * M_PI;
dAlpha + = 2.0 * M_PI;
}
} else
{
```