МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Дипломна магістерська робота

на тему

«Розробка математичного та програмного забезпечення для автоматизованого вводу інформації про зовнішні контури плоских об'єктів»

Виконав: студент групи МгІТ2-22

спеціальності 122 Комп'ютерні науки

Олександр НЕВМЕРЖИЦЬКИЙ

Науковий керівник д.т.н., проф. **Віктор ЧУПРИНКА**

Рецензент д.ф.-м.н.,проф. **Сергій КРАСНИТСЬКИЙ**

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
(повне найменування університету)

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
(назва факультету)

КАФЕДРА КОМП'ЮТЕРНИХ НАУК
(повна назва кафедри)

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

Володимир ШЕРБАНЬ

"____" _____ 2023 р.

З А В Д А Н Н Я

НА ДИПЛОМНУ МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Невмержицькому Олександру Анатолійовичу

1. Тема роботи «Розробка математичного та програмного забезпечення для автоматизованого вводу інформації про зовнішні контури плоских об'єктів»
науковий керівник роботи д.т.н., професор Чупринка Віктор Іванович**,**
затверджені наказом вищого навчального закладу від «12» вересня 2023 року, № 210-уч.
2. Строк подання студентом роботи - 10 листопада 2023 р.
3. Вихідні дані до роботи: розробка кафедри комп'ютерних наук, літературні джерела з конструювання та технології одягу
4. Зміст дипломної роботи (перелік питань, які потрібно розробити)
Вступ, РОЗДІЛ 1(Машинна графіка); РОЗДІЛ 2 (Математичне забезпечення для автоматизованого вводу інформації про зовнішні контури плоских об'єктів); РОЗДІЛ 3(Програмне забезпечення для автоматизованого вводу інформації про зовнішні контури плоских об'єктів)

5. Консультанти розділів дипломної магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Вступ	д.т.н., проф. Чупринка В.І.		
Розділ 1	д.т.н., проф. Чупринка В.І.		
Розділ 2	д.т.н., проф. Чупринка В.І.		
Розділ 3	д.т.н., проф. Чупринка В.І.		
Висновки	д.т.н., проф. Чупринка В.І.		

6. Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної магістерської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	05.09. 2023	
2	Розділ 1	20.09.2023	
3	Розділ 2	03.10.2023	
4	Розділ 3	25.10.2023	
5	Висновки	28.10.2023	
6	Оформлення дипломної магістерської роботи (чистовий варіант)	29.11.2023	
7	Здача дипломної магістерської роботи на кафедру для рецензування (за 14 днів до захисту)	31.10.2023	
8	Перевірка дипломної магістерської роботи на наявність ознак плагіату (за 10 днів до захисту)	01.10.2023	
9	Подання дипломної магістерської роботи у відділ магістратури для перевірки виконання додатку до індивідуального навчального плану (за 10 днів до захисту)	07.11.2023	
10	Подання дипломної магістерської роботи на затвердження завідувачу кафедри (з 7 днів до захисту)	9.12.2023	

Студент _____ **Олександр НЕВМЕРЖИЦЬКИЙ**
(підпис) (ім'я та прізвище)

Науковий керівник роботи _____ **Віктор ЧУПРИНКА**

АНОТАЦІЯ

Невмержицький О.А. Розробка математичного та програмного забезпечення для автоматизованого вводу інформації про зовнішні контури плоских об’єктів.

Дипломна магістерська робота за спеціальністю 122- «Комп’ютерні науки» – Київський національний університет технологій та дизайну, Київ, 2023 рік.

В роботі запропонований методи та алгоритми для автоматизованого вводу інформації про зовнішні контури плоских об’єктів. Запропоновані алгоритми реалізовані в програмний продукт для автоматизованого вводу інформації про зовнішні контури плоских об’єктів.

Розроблений програмний продукт має дружній інтерфейс та не потребує спеціальних знань з комп’ютерної техніки для роботи з ним.

Ключові слова: математичне та програмне забезпечення, зовнішні контури, плоскі об’єкти

ABSTRACT

Nevmerzhytskyi O.A. Development of Mathematical and Software for Automated Input of Information on External Contours of Flat Object.

Master's thesis in specialty 122- "Computer Science" – Kyiv National University of Technologies and Design, Kyiv, 2023.

The paper proposes methods and algorithms for automated input of information about the outer contours of plane objects**.** The proposed algorithms are implemented in a software product for automated input of information about the outer contours of flat objects.

The developed software product has a user-friendly interface and does not require special knowledge of computer technology to work with it.

Keywords: mathematical and software, outer contours, flat objects.

Зміст

ВСТУП

Легка промисловість одна із найбільших галузей легкої промисловості. Одним з основних завдань, які стоять перед працівниками швейної промисловості, є покращення якості та розширення асортименту продукції, що відповідає сучасним вимогам, на основі розвитку виробництва, підвищення ефективності за рахунок використання останніх досягнень науки та техніки.

На стадії проектування закладаються основи якості продукції, оскільки властивості продукції багато в чому залежить від процесу виготовлення Підготовка виробництва до випуску нових моделей виробів передбачає розробку технологічного процесу виготовлення. Ця робота має бути виконана в стислі терміни з мінімальними витратами, причому має бути обраний оптимальний варіант виробництва, тому що на стадії проектування технологічних процесів заздалегідь задаються інтенсивність функціонування підприємства, тобто. можливий рівень його техніко-економічних показників роботи.

З розвитком науково-технічного процесу значно розширився асортимент застосовуваних матеріалів, збільшилась номенклатура обладнання та засобів малої механізації. Зі збільшенням асортименту виробів виникла необхідність у скороченні термінів підготовки виробництва нових моделей виробів.

Скорочення термінів технологічних рішень та зменшення вартості підготовки виробництва може бути досягнуто не шляхом збільшення на підприємствах числа технологів, а шляхом створення інформаційно- обчислювальних систем для механізації та автоматизації процесів проектування. Використання обчислювальної техніки та впровадження систем автоматизованого проектування технологічних процесів дозволить покращити кількість технологічних рішень, що приймаються, скоротити терміни їх отримання, звільнити інженерно-технічних працівників від обчислювальної та технічної роботи для вирішення творчих завдань, спрямованих на вдосконалення

виробництва, створення нових, прогресивних методів обробки та технологічних процесів загалом.

Тому задача математичного та програмного забезпечення для автоматизованого вводу інформації про зовнішні контури деталей виробів легкої промисловості є актуальною, так як вона направлена на впровадження компютерних технологій у підготовчо-розкрійне виробництво.

Методи дослідження. Теоретичні дослідження роботи грунтуються на комплексі основних положень у підготовчо-розкрійне виробництва виробів легкої промисловості та застосуванні комп'ютерних наук при виришенні поставленої задачі.

Експерементальні дослідження заключалися в тестуванні розробленого продукту для проектування декоративних елементів на деталях взуття.

Наукова новизна полягає у розробці математичного та програмного забезпечення для автоматизованого вводу інформації про зовнішні контури плоских об’єктів.

Апробація результатів магістерської роботи. Основні положення і результати магістерської роботи протягом 2023 року були представлені та одержали позитивну оцінку на міжнародній науковій конференції.

Публікації. За темою магістерської роботи «Розробка математичного та програмного забезпечення для інтерактивної побудови щільних укладок плоских об’єктів» опубліковано одна наукових робота.

1.МАШИННА ГРАФІКА

1.1. Апаратні засоби машинної графіки

Математичне та програмне забезпечення комп'ютерної графіки не можна розглядати у відриві від апаратних засобів, що застосовуються на різних етапах роботи із зображеннями. Всі ці пристрої прийнято ділити на великі групи:

• пристрої введення (сканери, дігітайзери/графічні планшети, цифрові фото- та відеокамери);

• пристрої виведення (монітори, принтери, плотери, цифрові проектори);

• пристрої обробки (графічні прискорювачі, кодери MPEG та ін.).

Докладніше зупинимося тільки на апаратних засобах першої групи, на ринку спостерігається зараз бум нових засобів та технологій.

Пристрої введення

Існують різні технічні засоби, що здійснюють процес перетворення зображень на цифрову форму, наприклад: сканери, дігітайзери (графічні планшети), цифрові фото- та відеокамери. У кожному конкретному випадку важливо правильно вибрати потрібний пристрій, керуючись його технічними характеристиками, для отримання оцифрованого зображення з необхідною детальністю та гамою кольорів.

СКАНЕРИ. Сканер називається пристрій, що дозволяє вводити в комп'ютер образи зображень, представлених у вигляді тексту, малюнків, слайдів, фотографій або іншої графічної інформації. Традиційно сканери служили для вирішення спеціалізованих завдань: введення та запам'ятовування зображень у настільних видавничих системах, організації зберігання текстових документів у юридичних фірмах і т.п. подібно до принтерів і модемів.

Принцип дії та види сканерів. Принцип дії майже всіх типів сканерів єдиний. Він заснований на тому, що спрямованим променем висвітлюються окремі точки вихідного зображення (оригіналу) і відбитий в результаті промінь

сприймається фоточутливим приймачем, де інформація про колір точки інтерпретується як конкретне чисельне значення, яке через певний інтерфейс передається в комп'ютер. Як правило, світлочутливі елементи об'єднують у матрицю, щоб сканувати одночасно цілу ділянку оригіналу. За механізмом переміщення матриці світлочутливих елементів щодо оригіналу виділяють такі типи сканерів[1]:

1) *Планшетний сканер (Flatbed Scanner)* – сканер, у якому оригінал кладеться на скло та сканується за допомогою рухомої лінійної матриці (рис. 1.1). Розміри матриці та системи фокусування підібрані так, щоб вести сканування аркуша по всій ширині.

Рис.1.1. Планшетний сканер

2) *Ручний сканер (Handheld Scanner)* – портативний сканер, у якому сканування здійснюється шляхом ручного переміщення сканера оригіналом. За принципом дії такий сканер аналогічний до планшетного. Ширина області сканування трохи більше)

3) Барабанний сканер (Drum Scanner) - сканер, в якому оригінал закріплюється на барабані, що обертається. При цьому сканується точкова

область зображення, а головка, що сканує, рухається вздовж барабана на дуже маленькій відстані від оригіналу.

Основні характеристики. При виборі конкретної моделі сканера необхідно враховувати низку характеристик, пов'язаних із технічними можливостями моделі.

Роздільна здатність (Resolution) – кількість точок або растрових осередків, з яких формується зображення, на одиницю довжини чи площі. Чим більша роздільна здатність пристрою, тим дрібніші деталі можуть бути відтворені.

Апаратна/оптична роздільна здатність сканера (Hardware/optical Resolution) – це одна з основних характеристик сканера, безпосередньо пов'язана із щільністю розміщення чутливих елементів на матриці сканера. Вимірюється кількість пікселів на квадратний дюйм зображення – PPI (Pixel Per Inch). Приклад: 300×300 ppi.

Інтерпольоване дозвіл (Interpolated Resolution) – роздільна здатність зображення, отриманого за допомогою математичної обробки вихідного зображення. З покращенням якості має мало спільного. Часто служить рекламним прийомом для непідготовлених користувачів. Приклад: 600×1200 (9600) ppi (цифра 600 – максимальна оптична роздільна здатність, 1200 – роздільна здатність "подвійного кроку", 9600 – максимальна інтерпольована роздільна здатність).

Глибина кольору (color depth) – кількість розрядів кожного пікселя в цифровому зображенні, у тому числі сканером. Описує максимальну кількість кольорів, що відтворюється сканером у вигляді ступеня числа 2. Одному розряду відповідає чорно-біле зображення, 8-ми - сіре напівтонове, 16-ти - кольорове, 24-колірне зображення - найближче до людського сприйняття (модель RGB), 36bit і більше – повнокольорове зображення з високою достовірністю передачі кольорів, призначене для професійної роботи, найчастіше у видавничій справі.

Фірми-виробники На світовому ринку представлено досить велику кількість фірм-виробників сканерів. Найбільш популярні моделі виробляють Hewlett-Packard, Agfa, Canon, Mustek.

ДИГІТАЙЗЕРИ Дігітайзер (графічний планшет) – це пристрій, призначений для оцифрування зображень, що використовується для створення на комп'ютері малюнків та нарисів (рис. 1.2). Художник створює зображення на екрані, але його рука водить пером планшета. Як правило, планшет використовують професійні художники для точнішої обробки (створення) зображень.

Рис. 1.2. Графічний планшет

Крім того, дигітайзер можна використовувати просто як аналог маніпулятора "миша".

Принцип дії. Дігітайзер, або планшет, як його часто називають, складається з двох основних елементів: основи та курсора, що рухається його поверхнею. Принцип дії дигітайзера заснований на фіксації розташування курсору за допомогою вбудованої в планшет сітки. При натисканні на кнопку

курсора його розташування на поверхні планшета фіксується, а його координати передаються на комп'ютер. Сітка складається з дротяних або друкованих провідників із досить великою відстанню між сусідніми провідниками (від трьох до шести мм).

Основні характеристики Механізм реєстрації дозволяє отримати крок зчитування інформації набагато менше за крок сітки (до 100 ліній на мм). Крок зчитування інформації називається роздільною здатністю дигітайзера. Крок зчитування сітки, що реєструє, є фізичною межею дозволу дигітайзера. Ми говоримо про межу дозволу, тому що слід розрізняти роздільну здатність як характеристику приладу і програмно-задається дозвіл, що є змінна величина в налаштуванні дигітайзера.

Слід зазначити, що у роботі планшетів можливі перешкоди з боку випромінюючих пристроїв, зокрема моніторів. Незалежно від принципу реєстрації існує похибка у визначенні координат курсору, що називається точністю дигітайзера. Ця величина залежить від типу дигітайзера та від конструкції його складових. Точність існуючих планшетів коливається не більше від 0.005 дюйма до 0.03 дюйма.

Важливою характеристикою дигітайзера є число ступенів, що реєструється натискання електронного пера. У існуючих моделях ця величина може змінюватись у межах від одного до 256-ти. Програма-обробник використовує цю величину, встановлюючи залежно від неї, наприклад, товщину лінії (що сильніше натиск, тим товстіша лінія).

Фірми-виробники Найбільш популярними є моделі наступних фірм: CalComp, NUMONICS, WACOM.

ЦИФРОВІ ФОТОКАМЕРИ Цифрова камера отримує зображення, обробляє їх та зберігає у цифровому форматі. Замість плівки вона використовує вбудовану або змінну напівпровідникову пам'ять, щоб зберігати знімки. Вона має ті самі основні властивості, що й нормальна фотокамера, і, крім цього, може

з'єднуватися з комп'ютером, телевізором або принтером. Оскільки обробка кадру відбувається безпосередньо в камері, користувач може відразу перевірити правильність отриманого зображення, надрукувати його або надіслати електронною поштою.

Рис. 1.3. Цифрова фотокамера

Переваги цифрових фотокамер:

• Зображення обробляється відразу. Більшість цифрових камер мають маленький кольоровий екран, на якому можна негайно побачити зроблений знімок. Це дозволяє відмовитися від невдалого кадру та записати на його місце інший.

• Зображення зберігаються в електронній пам'яті, цикли запису-стирання інформації, можуть повторюватися практично нескінченно. Зникає потреба щоразу купувати плівку, реактиви на її прояви.

• Спростився процес введення фотографій у комп'ютер. Тепер не потрібно сканувати виготовлені звичайним чином фотографії. Ви просто підключаєте цифрову камеру за допомогою кабелю або PC-карти до ПК та переписуєте потрібні знімки на жорсткий диск.

• Цифрова камера дозволяє проводити безліч маніпуляцій із фотографіями. Недоліки цифрових фотокамер:

• Низька роздільна здатність. Прийнятною для якісного друку роздільною здатністю (понад 300 dpi) мають на сьогодні лише професійні цифрові камери з вартістю, що робить їх недоступними для масового користувача.

• Висока ціна, порівняно зі звичайними фотокамерами такого ж класу.

• Дійсно, якісний друк цифрових фотографій потребує найчастіше спеціального обладнання та має високу собівартість за рахунок дорогих витратних матеріалів.

Принцип дії Принцип дії цифрової фотокамери аналогічний принципу дії відеокамери і ось у чому. Пучок променів світла від об'єкта зйомки, проходячи через лінзу (або систему лінз) об'єктива та діафрагму, потрапляє на матрицю CCD (Charged Coupled Device). Матриця CCD або, як її ще називають, ПЗЗ (перетворювач світло-сигнал) є прямокутною матрицею зі світлочутливих елементів. Промінь світла, потрапляючи на чутливий елемент, перетворюється на аналоговий електричний сигнал. Аналогові сигнали від CCD перетворюються на цифрові, обробляються і записуються на згадку. Перетворення сигналів у цифрову форму здійснюється за допомогою аналого-цифрового перетворювача ADC.

Крім CCD, ADC та пам'яті в електричну схему цифрової фотокамери входять процесор DSP, який формує зображення з цифрових потоків, і конвертор JPEG, що стискає зображення для збільшення кількості кадрів, що зберігаються.

Змінна пам'ять використовується в цифрових камерах для збільшення кількості кадрів, що зберігаються і, найчастіше, являє собою Flash-карту пам'яті.

Фірми-виробники В даний час на ринку працюють десятки найвідоміших фірм-виробників як традиційного фотообладнання та матеріалів (Kodak, Konica, Nikon, Fuji, Agfa, Olympus та ін), так і комп'ютерної периферії та іншого

електронного обладнання (Hewlett-Packard, Seiko Epson, Sony, Ricoh, Mustek, UMAX, LG Electronics, Minolta та ін).

1.2. Графічні файлові формати

Як говорилося раніше, при зберіганні растрових зображень, зазвичай, доводиться мати справу з файлами великого розміру. У цьому важливою завданням є вибір відповідного формату файла.

Форматів графічних файлів існує безліч і вибір прийнятного не є очевидним завданням. Для полегшення вибору скористаємося класифікаціями.

За типом графічної інформації, що зберігається[2]:

• растрові (TIFF, GIF, BMP, JPEG);

• Векторні (AI, CDR, FH7, DXF);

• мішані/універсальні (EPS, PDF).

Слід враховувати, що файли практично будь-якого векторного формату дозволяють зберігати і растровую графіку. Однак часто це призводить до спотворень у передачі кольору, тому якщо зображення не містить векторних об'єктів, то краще використовувати растрові формати.

Скажімо кілька слів про найбільш популярні формати растрової графіки.

TIFF Запропонований компанією Aldus формат TIFF (Tagged Image File Format) на сьогоднішній день найближчий до статусу стандартного. Крім інших переваг формат TIFF дозволяє зберігати растрові зображення з компресією без втрати якості. Крім традиційних кольорів CMY формат підтримує кольороподіл із великою кількістю фарб, зокрема систему Hexahrome компанії Pantone. Цей формат підтримує стиск без втрати якості за алгоритмом LWZ-компресії. Найкращий для поліграфії. Принцип зберігання даних ґрунтується на використанні спеціальних маркерів (тегів) у поєднанні з бітовими послідовностями шматків растру.

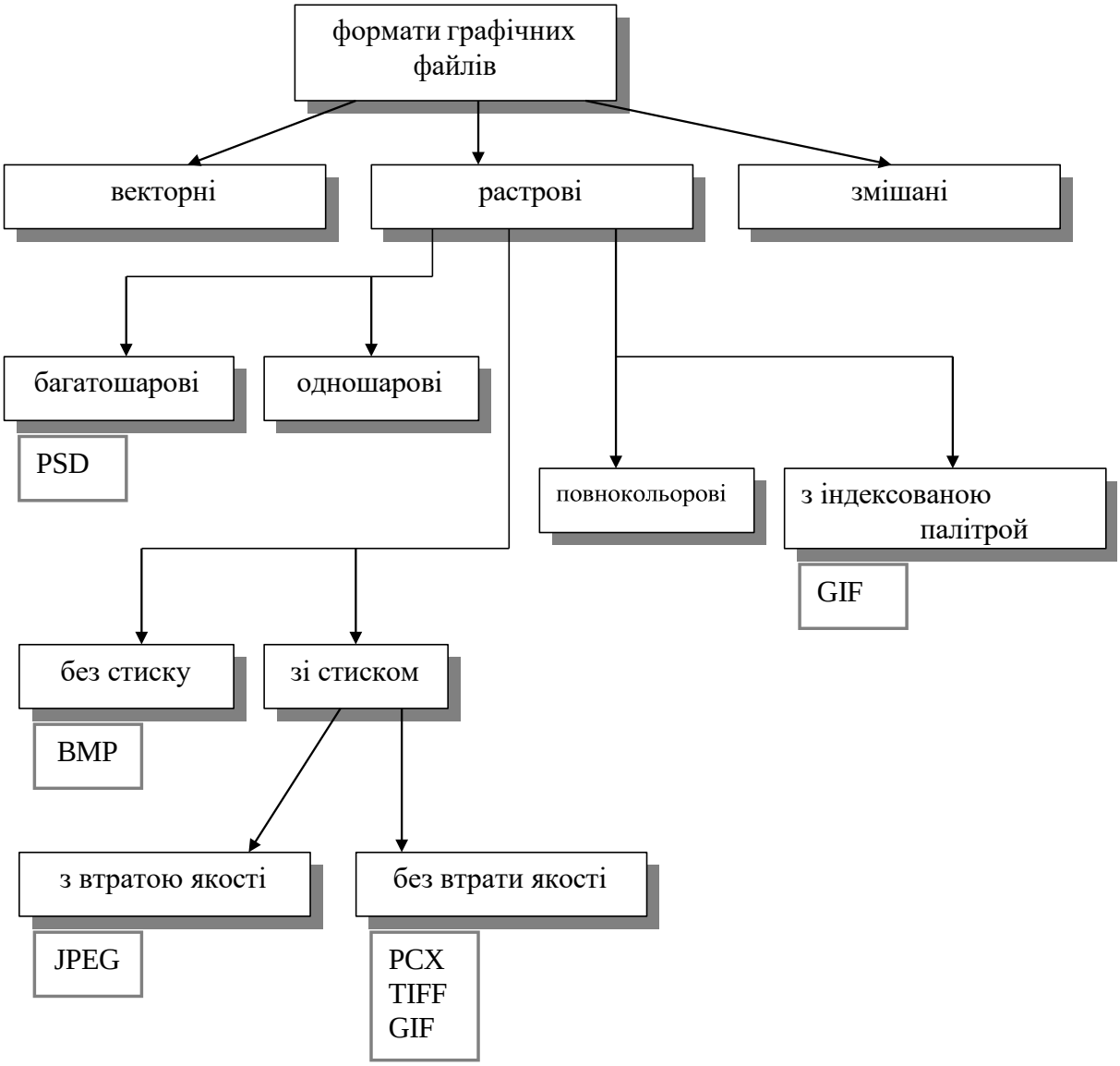


Рис. 5.4. Графічні файлові формати

GIF Перша версія формату GIF (Graphics Interchange Format, Формат для обміну графічною інформацією) була розроблена в 1987 р. фахівцями комп'ютерної мережі CompuServe. Цей формат поєднує в собі рідкісний набір переваг, неоціненних при тій ролі, яку він грає в WWW. Сам по собі формат містить досить добре упаковані графічні дані.

Як і в програм-архіваторів, ступінь стиснення графічної інформації в GIF залежить від рівня її повторюваності та передбачуваності, а іноді ще й від орієнтації картинки; оскільки GIF сканує зображення по рядках, то, наприклад, плавний перехід кольорів (градієнт), спрямований зверху вниз, стиснеться

набагато краще, ніж тих самих розмірів градієнт, орієнтований зліва направо, а останній – краще, ніж градієнт по діагоналі.

GIF може мати будь-яку кількість кольорів від двох до 256, і якщо в зображенні використовується, скажімо, 64 кольори (26), то для зберігання кожного пікселя буде використано рівно шість біт і ні бітом більше.

Змінивши порядок прямування даних у файлі, творці GIFa змусили картинку малюватись не тільки зверху вниз, а й, якщо можна так висловитися, «з глибини до поверхні», тобто ставати все чіткішими та детальнішими в міру підходу з мережі нових даних.

Для цього файл із зображенням тасується при записі так, щоб спочатку йшли всі рядки пікселів з номерами, кратними восьми (перший прохід), потім чотирма (другий прохід), потім двом і, нарешті, останній прохід – всі рядки, що залишилися, з непарними номерами. Під час прийому та декодування такого файлу кожен наступний прохід заповнює дірки в попередніх, поступово наближаючи зображення до вихідного стану. Тому такі зображення були названі черезрядковими (interlaced).

Іншою корисною можливістю формату є використання прозорості. Формат можна використовувати для створення анімаційних зображень. Для створення таких файлів використовується утиліта GIFConstractionSet.

Формат можна використовувати для створення анімаційних зображень. Для створення таких файлів використовується утиліта GIFConstractionSet.

BMP Формат BMP (від слова bitmap) був створений компанією Microsoft і широко використовується в операційних системах Windows для растрової графіки. Вам потрібно записати зображення в цьому форматі, якщо ви хочете використовувати його як фон вашого робочого столу. Хоча у цьому форматі може застосовуватися компресія, більшість програм її не використовують.

У файлах BMP інформація про колір кожного пікселя кодується 1, 4, 8, 16 або 24 біт (біт/піксел).

JPEG (Joint Photographic Experts Group) Строго кажучи, JPEG називається не формат, а алгоритм стиснення, заснований не так на пошуку однакових елементів, як і RLE і LZW, але в різниці між пікселями. JPEG шукає плавні переходи в квадратах 9×9 пікселів. Замість дійсних значень JPEG зберігає швидкість зміни від пікселя до пікселя. Зайву з його погляду колірну інформацію він відкидає, середня деякі значення. Чим вище рівень компресії, тим більше даних відкидається і тим нижча якість. Використовуючи JPEG, можна отримати файл у 10-500 разів менше, ніж BMP. Формат апаратно незалежний, повністю підтримується на PC та Macintosh, проте він відносно новий і не розуміється старими програмами (до 1995 р.). Зі сказаного можна зробити такі висновки. За допомогою JPEG краще стискаються растрові картинки фотографічної якості, ніж логотипи або схеми – у них більше півтонових переходів, серед однотонних заливок з'являються небажані перешкоди. Зображення з високою роздільною здатністю (200–300 і більше dpi) стискаються з меншими втратами, ніж із низьким (72–150 dpi), оскільки у кожному квадраті 9×9 пікселів переходи виходять м'які за рахунок того, що їх (квадратів) в файлах високої роздільної здатності більше. У форматі JPEG слід зберігати тільки кінцевий варіант роботи, тому що кожне перезбереження призводить до нових втрат (відкидання) даних і перетворення вихідного зображення в кашу. Як це не парадоксально, можливості алгоритму стиснення JPEG реалізовані у форматі JPEG не повністю.

PDF Формат PDF (Portable Document Format) запропонований фірмою Adobe як незалежний від платформи формат, в якому можуть бути збережені і ілюстрації (векторні та растрові), і текст, причому з безліччю шрифтів та гіпертекстових посилань. Для досягнення продекларованої в назві переносимості розмір PDF-файлу має бути малим. Для цього використовується компресія (до кожного виду об'єктів застосовується свій спосіб). Наприклад, растрові зображення записуються у форматі JPEG. Для роботи з цим форматом Adobe випустила пакет Acrobat. Безкоштовна утиліта Acrobat Reader дозволяє читати

документи та роздруковувати їх на принтері, але не дозволяє створювати або змінювати їх. Acrobat Distiller переводить у цей формат PostScript-файли. Багато програм (Adobe PageMaker, CorelDraw, FreeHand) дозволяють експортувати свої документи до PDF, а деякі – ще й редагувати графіку, записану в цьому форматі. Зазвичай у цьому форматі зберігаються документи, призначені лише для читання, але не для редагування. Файл PDF містить усі необхідні шрифти. Це зручно і дозволяє не передавати шрифти для виведення (передача шрифтів не є цілком законною з погляду авторського права).

PostScript Це мова опису сторінок, призначений для формування зображень довільної складності та виведення їх на друк. Для цього у мові є широкий набір графічних операторів, що використовуються у довільній комбінації. Усі графічні оператори мови, що формують зображення, можна поділити на три групи. Це:

• векторна графіка, що дозволяє малювати прямі лінії, дуги, криві довільного розміру, орієнтації, ширини, зафарбовувати площі будь-якого розміру, форми, кольору; колір для ліній або заливок може задаватись у будь-якому з колірних просторів мови; будь-який описаний мовою контур може бути межею кліпування зображення; контур кліпування задає межі зображення, що малюється;

• робота з текстом – для виведення тексту довільного розміру у різних гарнітурах, розміщуючи його з довільною орієнтацією у довільному місці сторінки; текст повністю інтегрований з графікою – всі текстові символи трактуються як графічні постаті та можуть оброблятися будь-яким із графічних операторів;

• растрові зображення дозволяють виводити на аркуші скановані малюнки чи фотографії з масштабуванням та орієнтацією, джерелом растру може бути як поточний файл, що містить програму на PostScript, і зовнішній; зчитування колірних шарів може вестись як з одного файлу, так і з кількох сепарованих. Зображення, що описується мовою PostScript, ніяк не залежить від роздільної

здатності вихідного пристрою та його колірної глибини (числа кольорів). Наближення до конкретних можливостей вихідного пристрою - це процес, не пов'язаний з описом зображення на мові PostScript, і виконується для кожного вихідного пристрою по-своєму. У цьому полягає пристрій-незалежність мови PostScript. Якість зображення визначається конкретним вихідним пристроєм, його фізичними обмеженнями.

Формування зображення на вихідному пристрої є двоступінчастим процесом: 1. Програма генерує пристрій незалежного зображення на мові PostScript. 2. Система обробки зображення (інтерпретатор) інтерпретує зображення (програму) та наближає його до характеристик конкретного вихідного пристрою.

Висновки до першого розділу

1. Роздільна здатність сучасних сканерів дозволяють скопіювати креслення деталей виробів легкої промисловості із заданою точністю. Звідси зявляється можливість застосувати ці сканери при автоматизованого вводу інформації про зовнішні контури деталей виробів легкої промисловості.

2. РОЗРОБКА МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗОВАНОГО ВВОДУ ІНФОРМАЦІЇ ПРО ЗОВНІШНІ КОНТУРИ ПЛОСКИХ ОБ'ЄКТІВ

2.1. Інтерактивна підготовка інформації про зовнішні контури

Контури деталей виробів легкої промисловості в більшості випадків мають складну конфігурацію, тому їх апроксимують більш простими кривими. Найбільше розповсюдження одержала кусково-лінійна апроксимація, тобто апроксимація кривими першого порядку (прямими). Це зручно також для автоматизованого вводу такої інформації при використанні спеціальних пристроїв.

Нехай S - фігура з заданою орієнтацією. Зв'яжемо з деталлю S

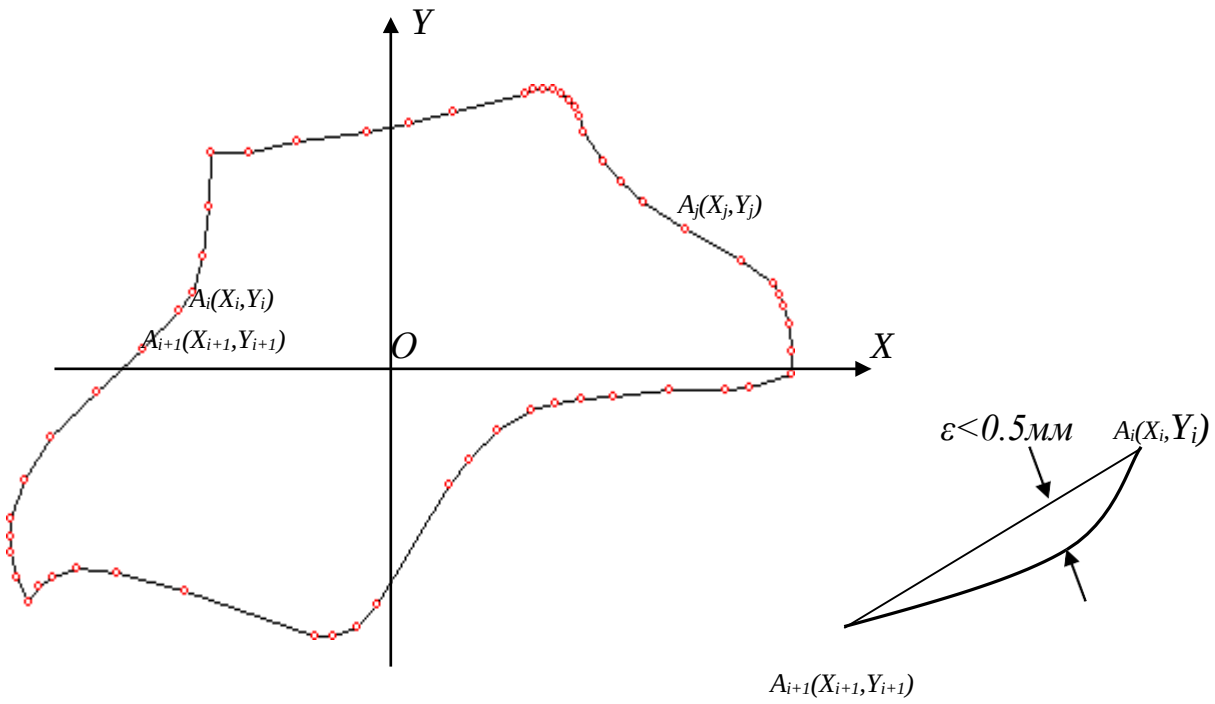


Рис. 2.1. Кусково-лінійна апроксимація

координатну систему XOY, де O – полюс деталі, обраний в будь-якій її внутрішній точці. Контур взуттєвої деталі S апроксимуємо ламаною лінією з вершинами в послідовно вибраних на контурі деталі точках (рис. 2.1.).

Будь-яка деталь S може бути представлена координатами вершин опукло-ввігнутого многокутника, тобто масивом $\{X_j, Y_j\}$,. $j=1..n$, а координати будь-якої точки на прямій між вершинами A_j та A_{j+1} можна визначити наступним чином [3]

$$\begin{aligned} X &= X_j + (X_{j+1} - X_j)t \\ Y &= Y_j + (Y_{j+1} - Y_j)t \\ 0 \le t \le 1;\ j &= 1,2..n \end{aligned} \qquad (2.1)$$

де j– порядковий номер відрізка між вершинами A_j та A_{j+1} ;

n– кількість точок апроксимації для деталі S;

t– параметр, що визначає положення точки на прямій між вершинами A_j та A_{j+1}.

Постановка задачі. Нехай маємо файл формату **.bmp* із кресленнями деталей виробів легкої промисловості. Необхідно отримати інформацію про зовнішні контури цих деталей у вигляді координат апроксимуючого многокутника; інформацію про деталі представити у файлі формату **.dgt* системи проектування моделей взуття «Ірис»[4].

В будь-якому файлі формату **.bmp* [5] в перших двох (0-1) байтах зберігаються символи 4D42h, які є буквами 'BM'; в наступних чотирьох (2-5) байтах зберігається розмір файлу в байтах; наступні чотири(6-9) байти – резервні поля; наступні чотири(10-13) байти – зсув, з якого починається саме зображення (растр); наступні чотири (14-17) байти – розмір заголовка BITMAP у байтах (дорівнює 40); наступні чотири (18-21) байти - ширина зображення в пікселах; наступні чотири (22-25) байти - висота зображення в пікселах; наступні два (26-27) байти - кількість площин, повинно бути 1; наступні два (28-29) байти – кількість біт на відображення одного піксела: 1, 4, 8 або 24; наступні чотири (30-

33) байти - тип стиску; наступні чотири (34-37) байти - 0 або розмір стиснутого зображення в байтах; наступні чотири (38-41) байти - горизонтальна роздільна здатність, *піксел/м*; наступні чотири (42-47) байти - вертикальна роздільна здатність, *піксел/м*; наступні чотири (46-49) байти - кількість використовуваних кольорів; наступні чотири(50-53) байти - кількість "важливих" кольорів. Далі йде інформація про пікселі самого зображення по рядкам.

Необхідно зауважити, що якщо інформація записана в декількох байтах, то перший байт є наймолодшим. Для виводу креслення на екран монітору нас будуть цікавити 18 -21 байти - ширина зображення в пікселах та 22-25 байти - висота зображення в пікселах.

Для визначення реальних координат вершин апроксимуючих многокутників для зовнішніх контурів деталей взуття, креслення яких представлені у файлі формату **.bmp*, нас будуть цікавити 38-41 байти - горизонтальна роздільна здатність в *піксел/мм* та 42-47 байти - вертикальна роздільна здатність в *піксел/мм*.

Нехай нам потрібно визначити ширину зображення в пікселах *ShIm*, що зберігається в 18-21 байтах, та значення 18, 19, 20, 21 байтів, що відповідно позначимо: *A,B,C,D*.

Тоді *ShIm* можна визначити за наступною формулою:

$$ShIm = A + 256 * B + 256^2 * C + 256^3 * D \qquad (2.2)$$

Аналогічно визначається висота зображення, горизонтальна та вертикальна роздільна здатність в *піксел/м*.

Нехай координати вершин апроксимуючого многокутника для зовнішнього контуру деталі взуття в пікселах представлено масивом $\{X_j^E, Y_j^E\}$, *j=1..n*. Тоді реальні координати вершин в *мм* можна розрахувати за наступними формулами[6]:

$$\begin{matrix} X_j = mx \cdot X_j^E \\ Y_j = my \cdot Y_j^E \end{matrix}, \quad \text{де} \quad \begin{matrix} mx = 1000/RSh \\ my = 1000/RVs \end{matrix}, \quad j = 1,2,...n \qquad (2.3),$$

де *RSh*- горизонтальна роздільна здатність в *піксел/м*,

RVs - вертикальна роздільна здатність в *піксел/м*.

На основі формул (2.1 -2.3) розроблений алгоритм інтерактивної підготовки інформації про зовнішні контури деталей виробів легкої промисловості.

Структурна схема алгоритму представлена на рис. 2.2.

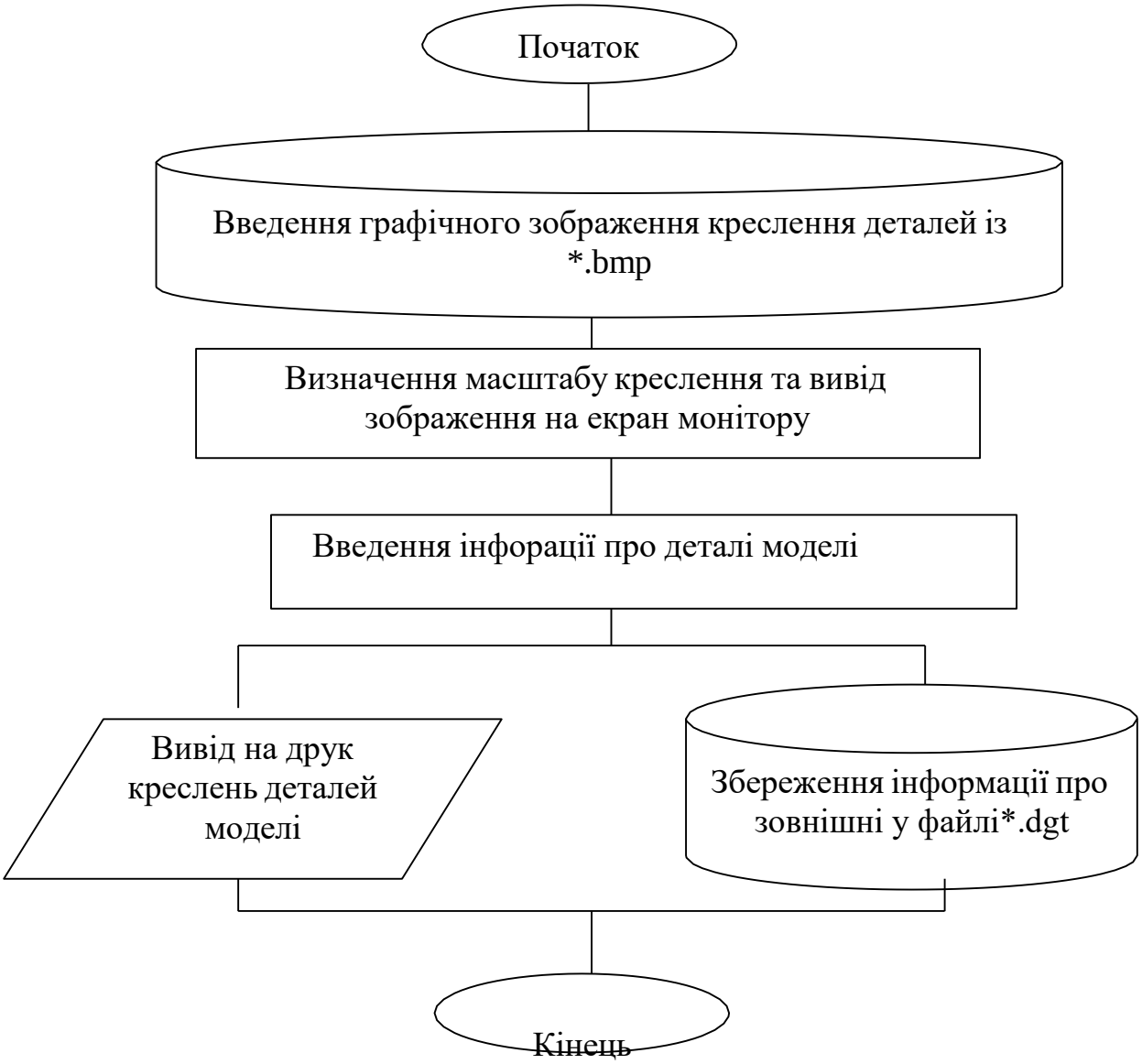


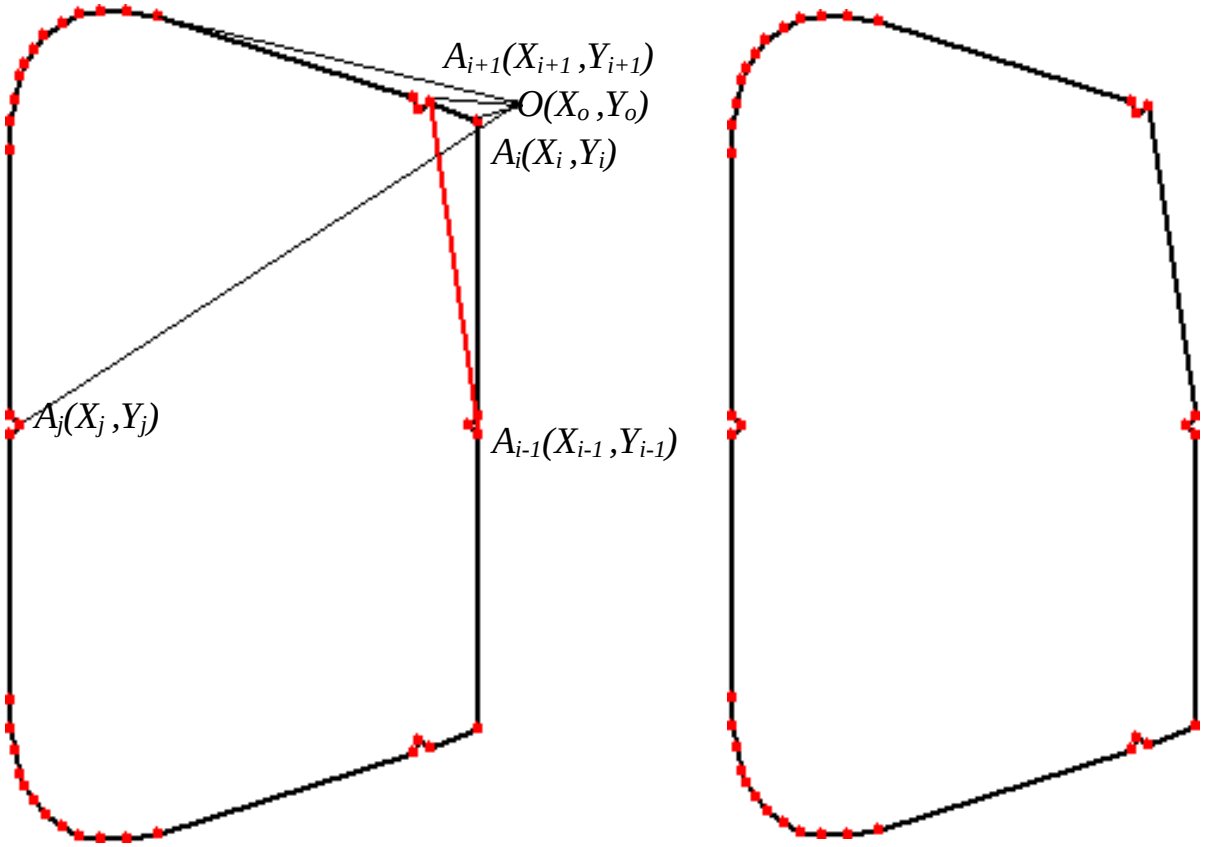
Рис. 2.2. Структурна схема алгоритму інтерактивної підготовки інформації про зовнішні контури деталей взуття

2.2 Алгоритм вилучення будь-якої вершини на контурі деталі

Нехай контур деталі представляє собою многокутник з вершинами $A_j(X_j,Y_j)$, $j=1..n$. Курсор знаходиться в точці $O(X_0,Y_0)$. Будемо вважати, що нам необхідно вилучити ту вершину, яка найменше віддалена від точки $O(X_0,Y_0)$ (рис.2.3).

Для вилучення необхідної вершини потрібно[7]:

- ідентифікувати необхідну вершину. Для цього потрібно виконати наступні дії:



а) *б)*

Рис.2.3. Вилучення вершини на контурі деталі

а) деталь до вилучення вершини б) деталь після вилучення вершини

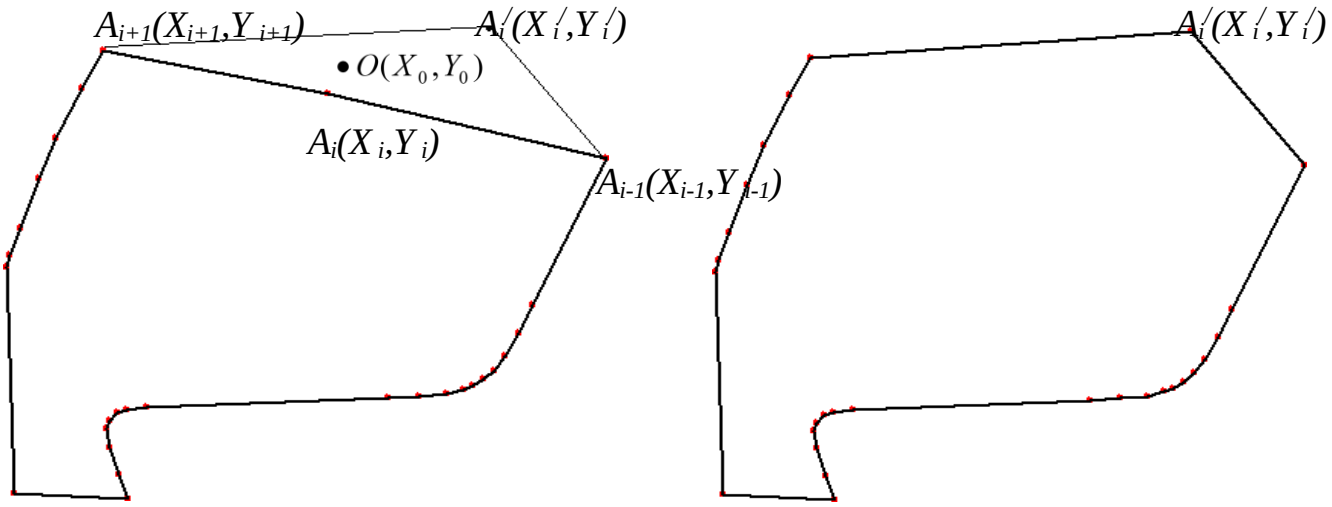
а) знайти відстані [8] $D_j = \sqrt{(Xo - X_j)^2 + (Yo - Y_j)^2}$ від точки $O(X_0,Y_0)$ до вершин $A_j(X_j,Y_j)$, $j=1..n$, $XoF=XcF+(Xo-XcE)/mxy$ та $YoF=YcF-(Yo-YcE)/mxy$;

b) визначити вершину A_i, для якої $D_i = \min\{D_j\}, j = 1..n$;

- вилучити цю вершину із файлу з інформацією про деталі;
- вилучити відрізки A_iA_{i-1} та A_jA_{j+1} на екрані;
- провести новий відрізок $A_{i+1}A_{i-1}$.

2.3. Алгоритм зміни положення будь-якої вершини на контурі деталі

Нехай контур деталі представляє собою многокутник з вершинами $A_j(X_j,Y_j)$, $j=1..n$. Курсор знаходиться в точці $O(X_0,Y_0)$. Будемо вважати, що нам необхідно вершину A_i, яка найменше віддалена від точки $O(X_0,Y_0)$(рис.2.4), перемістити в точку A_i'.



b) *б)*

Рис.2.4. Зміна положення вершини на контурі деталі

а*) деталь до зміни положення вершини на контурі деталі* *б) деталь після зміни положення вершини на контурі деталі*

Для зміни значення координат будь-якої вершини на контурі деталі потрібно:

- ідентифікувати необхідну вершину. Для цього потрібно виконати наступні дії:

  а) знайти відстані $D_j = \sqrt{(XoF - X_j)^2 + (YoF - Y_j)^2}$ від точки $O(X_0,Y_0)$ до вершин $A_j(X_j,Y_j)$, $j=1..n$,

  $XoF=XcF+(Xo-XcE)/mxy$ та $YoF=YcF-(Yo-YcE)/mxy$;

  b) визначити вершину A_i, для якої $D_i = \min\{D_j\}, j = 1..n$;

- знайти нове значення координат цієї вершини, тобто координати точки $A_i^{/}(X_i^{/},Y_i^{/})$ [8]:

$$X_i^{/} = XcF + (Xo^{/} - XcE)/mxy$$
$$Y_i^{/} = YcF - (Yo^{/} - YcE)/mxy \quad ,$$

де $O^{/}(X_0^{/},Y_0^{/})$- нове положення курсору (координати точки $A_i^{/}$ в системі координат, яка пов'язана з монітором;

- замінити у файлі з інформацією про деталі старі координати цієї вершини $A_i(X_i,Y_i)$ на нові $A_i^{/}(X_i^{/},Y_i^{/})$;

4. вилучити відрізки A_iA_{i-1} та A_jA_{j+1} на екрані;
5. провести нові відрізки $A_{i-1}A_i^{/}$ та $A_{i+1}A_i^{/}$.

2.4. Алгоритм введення додаткової вершини на контурі деталі

Нехай контур деталі представляє собою многокутник з вершинами $A_j(x_j,y_j)$, $j=1..n$. Курсор знаходиться в точці $O(X_0,Y_0)$. Будемо вважати, що нам необхідно між вершинами A_i та A_{i+1} ввести нову вершину $A^*(x^*,y^*)$, якщо точка $O(X_0,Y_0)$ (рис.2.5) найменше віддалена від сторони A_iA_{i+1} та проекція точки $O(X_0,Y_0)$ на пряму, яка проходить через точки A_i та A_{i+1} , буде належати цій стороні.

Для введення додаткової вершини на контурі деталі потрібно]9]:

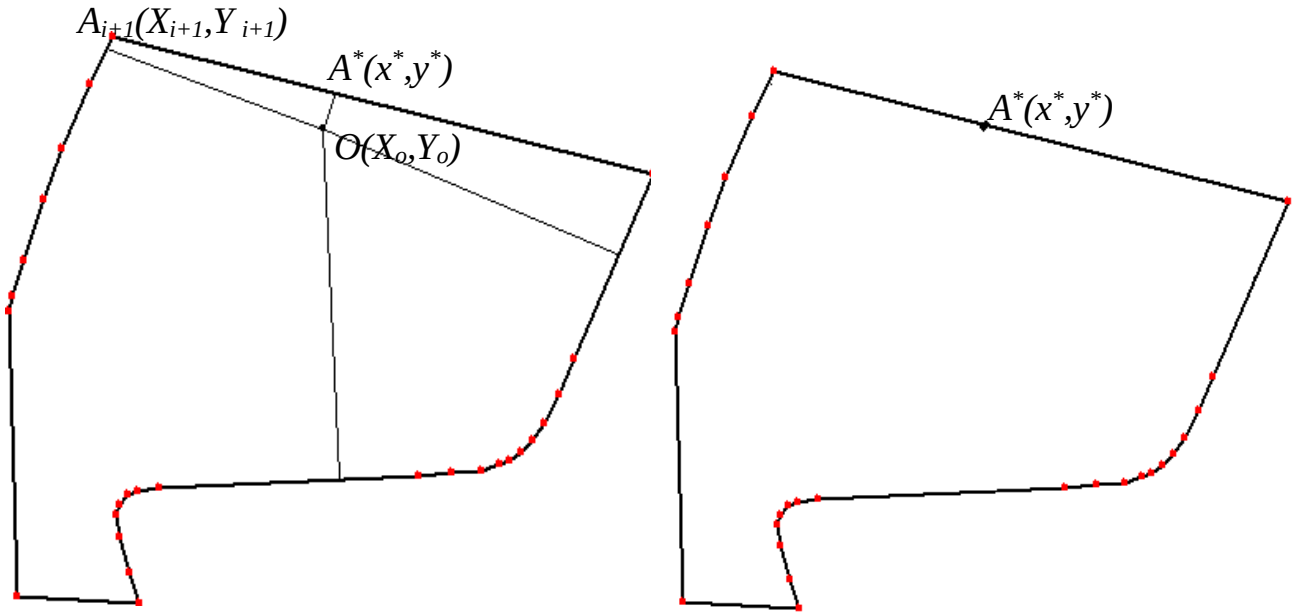
- ідентифікувати необхідну сторону $A_i A_{i+1}$ на зовнішньому контуру деталі.
- Для цього потрібно виконати наступні дії:

а) знайти відстані D_j точки $O(X_0,Y_0)$ до сторін A_jA_{j+1}, $j=1..n-1$,

де $D_j = A_j^* YoF + B_j^* XoF + C_j^*$, $XoF=XcF+(Xo-XcE)/mxy$ та

$YoF=YcF-(Yo-YcE)/mxy$; $A^*_j = (Y_{j+1} - Y_j)/Q_j$, $B^*_j = (Y_j - Y_{j+1})/Q_j$,

$C^*_j = ((X_{j+1} - X_j)Y_j - (Y_{j+1} - Y_j)Y_j)/Q_j$ та $Q_j = \sqrt{(X_{j+1} - X_j)^2 + (X_{j+1} - Y_j)^2}$.



а) *б)*

Рис. 2.5. Введення додаткової вершини на контурі деталі

а) *деталь до введення додаткової вершини* *б) деталь після введення додаткової вершини*

При пошуку D_j ми розглядаємо тільки ті сторони $A_j A_{j+1}$, для яких проекція точки $O(X_0,Y_0)$ на пряму, що проходить через точки A_j та A_{j+1} буде належати даному відрізку.

b) визначити відрізок $A_i A_{i+1}$, для якого $D_i = \min\{D_j\}, j = 1..n-1$;

- знайти значення координат нової вершини A^*, тобто координати точки $A^*(x^*,y^*)$;
- додати у файл з інформацією про деталі координати нової вершини $A^*(x^*,y^*)$;
- вилучити відрізок A_jA_{j+1} на екрані;
- провести нові відрізки $A_i A^*$ та A^*A_{i+1}.

Більш детально зупинимося на знаходженні відстані D від точки C до відрізка AB, якщо перпендикуляр, проведений із точки C на відрізок AB перетинає

його. Нехай точки *A, B, C* мають координати: $A(X_a,Y_a)$, $B(X_b,Y_b)$, $C(X_c,Y_c)$. Тоді рівняння прямої, що проходить через точки *A, B*, має вигляд[11]:

$$\frac{x-X_a}{X_b-X_a}=\frac{y-Y_a}{Y_b-Y_a}, \text{ або } Ax+By+C=0, \qquad (2.4)$$

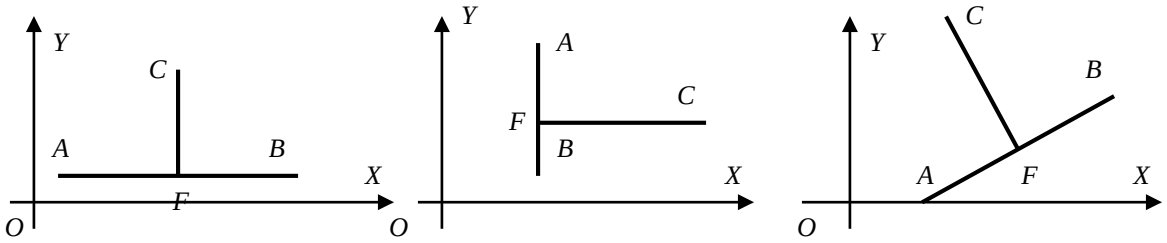
де $A=Y_b-Y_a;\ B=X_a-X_b;\ C=X_bY_a-X_aY_b.$

Відстань від точки *C* до прямої, що проходить через точки *A, B* визначається наступним чином [243]:

$$D=\frac{AX_c+BY_c+C}{\sqrt{A^2+B^2}} \qquad (2.5)$$

Знайдемо точку перетину перпендикуляра $F(X_f,Y_f)$, проведеного із точки *C* до прямої, що проходить через точки *A, B* з цією прямою. Можливі три випадки:

- *A=0* (пряма, що проходить через точки *A, B,* паралельна вісі *OX* (рис. 2.6.а). Тоді $X_f=X_c$, $Y_f=Y_a$.
- *B=0* (пряма, що проходить через точки *A, B,* паралельна вісі *OY* (рис. 2.6.б). Тоді $X_f=X_a$, $Y_f=Y_c$



а) *б)* *в)*

Рис. 2.6

- $A\neq 0$ та $B\neq 0$ (рис. 2.6.в). Тоді точку перетину перпендикуляра $F(X_f,Y_f)$, проведеного із точки *C* до прямої, що проходить через точки *A, B* з цією прямою знаходимо із наступної системи рівнянь:

$$\begin{cases} Ax + By + C = 0 \\ y - Y_c = K(x - X_c) \end{cases}, \quad \text{де} \quad K = -\frac{X_b - X_a}{Y_b - Y_a} = \frac{X_a - X_b}{Y_a - Y_b} \qquad (2.6)$$

Розв'язавши систему рівнянь (2.20) знайдемо координати точки перетину *F*:

$$\begin{aligned} X_f &= \frac{K \cdot B \cdot X_c - B \cdot Y_c - C}{A + K \cdot B}, \\ Y_f &= Y_c + K \cdot (X_f - X_c) \end{aligned} \quad . \qquad (2.7)$$

Тепер нам залишилося впевнитись, що точка перетину *F* лежить всередині відрізка *AB*. Нехай $T_1 = \min(X_a, X_b)$, $T_2 = \max(X_a, X_b)$, $Q_1 = \min(Y_a, Y_b)$ та $Q_2 = \max(Y_a, Y_b)$.

Тоді, якщо виконується наступна умова:

$$\begin{cases} T_1 \le X_f \le T_2 \\ Q_1 \le Y_f \le Q_2 \end{cases}, \qquad (2.8),$$

то точка перетину *F* лежить всередині відрізка *AB*.

Описану вище процедуру проробимо для всіх сторін зовнішнього контуру деталі. Серед тих сторін, для яких точка перетину перпендикуляра $F(X_f,Y_f)$, проведеного із точки *C* лежить на стороні, будемо шукати сторону, що найменше віддалена від точки *C*.

Висновки до другого розділу

1. Запропоновані алгоритми для для автоматизованого введення та інтерактивного коригування інформації про зовнішні контури плоских геометричних обєктів, які є базою для розробки необхідного програмного продукту.

3.РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ДЛЯ АВТОМАТИЗОВАНОГО ВВОДУ ІНФОРМАЦІЇ ПРО ЗОВНІШНІ КОНТУРИ ПЛОСКИХ ОБ'ЄКТІВ

3.1. Вимоги до програмного продукту

Програмний продукт повинен забезпечувати[12]

- надійну роботу як в локальному, так і в мережевому варіантах;
- високу точність. Будь-які обмеження на кількість внутрішніх контурів і число точок лекальних кривих в конструюванні ведуть, в кінцевому підсумку, до втрати точності при відтворенні складних деталей;
- гнучкість роботи. Як мінімум, повинні бути можливість скасування операцій на будь-яку кількість кроків, можливості введення та редагування будь-якої кількості додаткових точок і інших елементів креслення на будь-якому етапі конструювання. Дуже корисний механізм автоматичних прив'язок до характерних точок лекальні кривих;
- швидкість. Швидка змінюваність моделей, розширення асортиментної бази неможливі без потужного графічного редактора та конструкторського модуля (не плутати з «креслярськими засобами для конструювання»). Сучасний конструкторський модуль повинен забезпечувати виготовлення комплекту лекал для найскладнішої моделі протягом 2-3 годин;
- багато документальний інтерфейс. Сучасні системи дозволяють відкривати відразу кілька моделей при роботі. Вільно і наочно виділяти і переносити з моделі в модель будь-які елементи креслення - будь то лекала або окремі модельні лінії. Без обмеження комбінувати нові моделі на основі наявних;

- роботу з будь-яким серійним обладнанням. Вільний обмін даними з іншими програмами. Крім усього іншого, це полегшить створення єдиної мережі підприємства;
- вивід на друк в будь-якому масштабі на будь-якому етапі роботи.

3.2. Вибір мови програмування для практичної реалізації запропонованих методів та алгоритмів для автоматизований ввід інформації про зовнішні контури плоских об’єктів

C++ - компілювана, статично типізована мова програмування загального призначення. Підтримує такі парадигми програмування, як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування [13].

Мова має багату стандартну бібліотеку, яка включає поширені контейнери і алгоритми, введення-виведення, регулярні висловлювання, підтримку багатопоточності та інші можливості. C++ поєднує властивості як високорівневих, і низькорівневих мов програмування.

У порівнянні з його попередником - мовою C - найбільшу увагу приділено підтримці об'єктно-орієнтованого та узагальненого програмування. C++ широко використовується для розробки програмного забезпечення, будучи однією з найпопулярніших мов програмування. Область його застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для систем, високопродуктивних серверів, а також комп'ютерних ігор.

Існує безліч реалізацій мови C++ як безкоштовних, так і комерційних і для різних платформ. Наприклад, на платформі x86 це GCC, Clang, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder та інші. C++ вплинув інші мови програмування, насамперед Java і C#.

Синтаксис C++ успадкований від мови C. Спочатку одним із принципів розробки було збереження сумісності з C. Тим не менш, C++ не є в строгому сенсі надмножиною C; безліч програм, які можуть однаково успішно транслюватися як компіляторами C, і компіляторами C++, досить велике, але включає всі можливі програми на C.

3.3. Опис основних процедур розробленого програмного продукту, які забезпечують автоматизований ввід інформації про зовнішні контури плоских об'єктів

Функція *void __fastcall TForm1::Bmp1Click(TObject *Sender)* забезпечує введення зображення креслень плоских об'єктів у форматі *. *Bmp* на *Image1* та визначення роздільної здатності зображення по вісі ОХ та вісі ОУ[14-16]:

```
void __fastcall TForm1::Bmp1Click(TObject *Sender)
{
  unsigned int A,B,C,D;
int i,L,DlIm, ShIm,RzDl,RzSh;
float Xr[100],Yr[100];
char  Fname[100],T;
AnsiString NameF;

if (OpenPictureDialog1->Execute())
  Image1->Picture->LoadFromFile(OpenPictureDialog1->FileName);
  NameF=OpenPictureDialog1->FileName;
   for (i=0;i<99;i++)Fname[i]=NULL;
//   ifstream f;
   L=NameF.Length();
   for(i=1; i<=L;i++)
   {
    T=NameF[i];
    Fname[i-1]=T;
   }
   f.open(Fname,ios::in);
    DlIm=Val(18);
    ShIm=Val(22);
```

```
   RzDl=Val(38);
  RzSh=Val(42);
  f.close();
  mx=1000./RzDl;
  my=1000./RzSh;
  Image1->Width=DlIm;
  Image1->Height=ShIm;
}
```

Функція *void __fastcall TForm1::Image1MouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y)* забезпечує інтерактивне введення інформації про зовнішні контури креслень плоских об’єктів.

```
void __fastcall TForm1::Image1MouseDown(TObject *Sender,
     TMouseButton Button, TShiftState Shift, int X, int Y)
{
  int Xr,Yr;
 Image1->Canvas->Pen->Mode=pmXor;
 Image1->Canvas->Pen->Color=clGreen;
 Image1->Canvas->Pen->Width=2;
 if (Button==mbLeft)
  {
   Xd[p][i]=X*mx; Xt[i]=X;
   Yd[p][i]=-Y*my; Yt[i]=Y;
   if (i==0)Image1->Canvas->MoveTo(Xt[i],Yt[i]);
   else Image1->Canvas->LineTo(Xt[i],Yt[i]);
   i++;
  }
 else
  if (Button==mbRight)
   {
    if (i>0)
     {
      i--;
      Image1->Canvas->LineTo(Xt[i],Yt[i]);
     }
   }
else
   {
    // i++;
     Xd[p][i]=Xd[p][0];
```

```
        Yd[p][i]=Yd[p][0];
        Image1->Canvas->LineTo(Xt[0],Yt[0]);
        p=p+1;
        if (p==1)  Model="AAAAA";
        NameDet[p-1]=IntToStr(p);
        KilksPointDet[p-1]=i+1;
        KilDet=p;
        i=0;
        }
    }
```

Функція *void GraphIm2(int n, float X[], float Y[], float Xcf, float Ycf, float Xce, float Yce, float mxy,int q, int p)* забезпечує вивід зображення многокутника на область *Image2*.

```
    void GraphIm2(int n, float X[], float Y[], float Xcf, float Ycf,
      float Xce, float Yce, float mxy,int q, int p)
{
 int j,Xr[300],Yr[300];
 for(j=0;j<n;j++)
  {
   Xr[j]=floor((X[j]-Xcf)*mxy+Xce);
   Yr[j]=floor(-(Y[j]-Ycf)*mxy+Yce);
  }
Form1->Image2->Canvas->Pen->Width=p;
Form1->Image2->Canvas->Pen->Mode=pmCopy;
switch(q)
  {
   case 1:Form1->Image2->Canvas->Pen->Color=clRed;break;
   case 2:Form1->Image2->Canvas->Pen->Color=clBlue;break;
   case 3:Form1->Image2->Canvas->Pen->Color=clGreen;break;
   case 4:Form1->Image2->Canvas->Pen->Color=clGray;break;
   default:Form1->Image2->Canvas->Pen->Color=clBlack;
  }
 for(j=0;j<n;j++)
  if(j==0)Form1->Image2->Canvas->MoveTo(Xr[j],Yr[j]);
  else Form1->Image2->Canvas->LineTo(Xr[j],Yr[j]);
}
```

Функція *int NumPointDet(int n, float X[], float Y[], float XcurF, float YcurF)*

визначає порядковий номер активної вершини для многокутника з кількістю вершин *int n* та координатами вершин *float X[], float Y[]*.

```
int NumPointDet(int n, float X[], float Y[], float XcurF, float YcurF)
 {
 int i,p;
 float Dmin,D;
 p=0;
 Dmin=sqrt(powl(X[0]-XcurF,2)+powl(Y[0]-YcurF,2));
 for(i=1;i<n;i++)
  {
   D=sqrt(powl(X[i]-XcurF,2)+powl(Y[i]-YcurF,2));
   if(Dmin>D)
    {
     p=i;
     Dmin=D;
    }
   }
   return p;
}
```

Функція *void DelPoint(int &n, float X[],float Y[],int NP)* вилучає активну вершину[17-18] з порядковим номером *int NP* у многокутнику з кількістю вершин *int n* та координатами вершин *float X[], float Y[]*.

```
void DelPoint(int &n, float X[],float Y[],int NP)
{
 int i;
  if (NP==n-1) NP=0;
   for(i=NP;i<n-1;i++)
   {
    X[i]=X[i+1];
    Y[i]=Y[i+1];
    }
  if (NP==0)
   {
     X[n-2]=X[0];
     Y[n-2]=Y[0];
    }
    n--;
```

```
}
```

Функція *void ProjectionPointOnLine(float Xa, float Ya, float Xb, float Yb, floatXc, float Yc, float& Xo,float& Yo)* визначає координати *float& Xo,float& Yo* проекції точки з координатами *floatXc, float Yc* на пряму, яка проходить через точки *(float Xa, float Ya, float Xb, float Yb.*

```
void ProjectionPointOnLine(float Xa,float Ya,float Xb,float Yb,float Xc,float Yc,
{
    float A,B,C,k;
     if (Ya==Yb)
      {
        Xo=Xc; Yo=Ya;
      }
     else if (Xa==Xb)
      {
        Xo=Xa; Yo=Yc;
      }
         else
          {
           A=Yb-Ya;
           B=Xa-Xb;
           C=Xb*Ya-Xa*Yb;
           k=-A/B;
           Xo=(C/B+Xc/k+Yc)/(k+1/k);
           Yo=k*Xo-C/B;
          }
}
```

Функція *bool PointBelongsLineSegment(float Xa,float Ya,float Xb,float Yb,float Xc,float Yc, float Xo,float Yo,float& D)* визначає належить проекція *float Xo,float Yo* точки *float Xc,float Yc* відрізку з координатами вершин *float Xa,float Ya,float Xb,float Yb* та відстань *float& D* точки з координатами *float Xc,float Yc* до цього відрізку

```
bool PointBelongsLineSegment(float Xa,float Ya,float Xb,float Yb,float Xc,float Yc, float Xo,float Yo,float& D)
 {
   float T1,T2,Q1,Q2;
```

```
if (Xa<Xb)
 {
  T1=Xa; T2=Xb;
  }
 else
  {
  T1=Xb; T2=Xa;
  }
if (Ya<Yb)
 {
  Q1=Ya; Q2=Yb;
  }
 else
  {
  Q1=Yb; Q2=Ya;
  }
  if (Xo>=T1&&Xo<=T2&&Yo>=Q1&&Yo<=Q2)
   {
    D=sqrt(pow(Xc-Xo,2)+pow(Yc-Yo,2));
    return True;
    }
   else
    {
    D=0;
    return False;
    }
}
```

Функція *void AddPoint(int &n, float X[],float Y[],float Xo, float Yo,int NVidr)* додає нову точку зкоординатами *float Xo, float Yo* у многокутник з кількістю вершин *int &n*, та координатами вершин, *float X[],float Y[]*, на стороні з порядковим номером *int NVidr*[19-20]

```
void AddPoint(int &n, float X[],float Y[],float Xo, float Yo,int NVidr)
 {
  int i;
   for(i=n-1;i>=NVidr+1;i--)
    {
```

```
    X[i+1]=X[i];
    Y[i+1]=Y[i];
    }
   X[NVidr+1]=Xo;
   Y[NVidr+1]=Yo;
   n++;
}
```

3.4. Інструкції по роботі з програмним продуктом

Початок роботи з програмою розпочинається з запуску файлу *PrNevmer.exe*. При цьому на екрані з‘являється головне вікно програми прийме наступний вигляд(рис. 3.1).

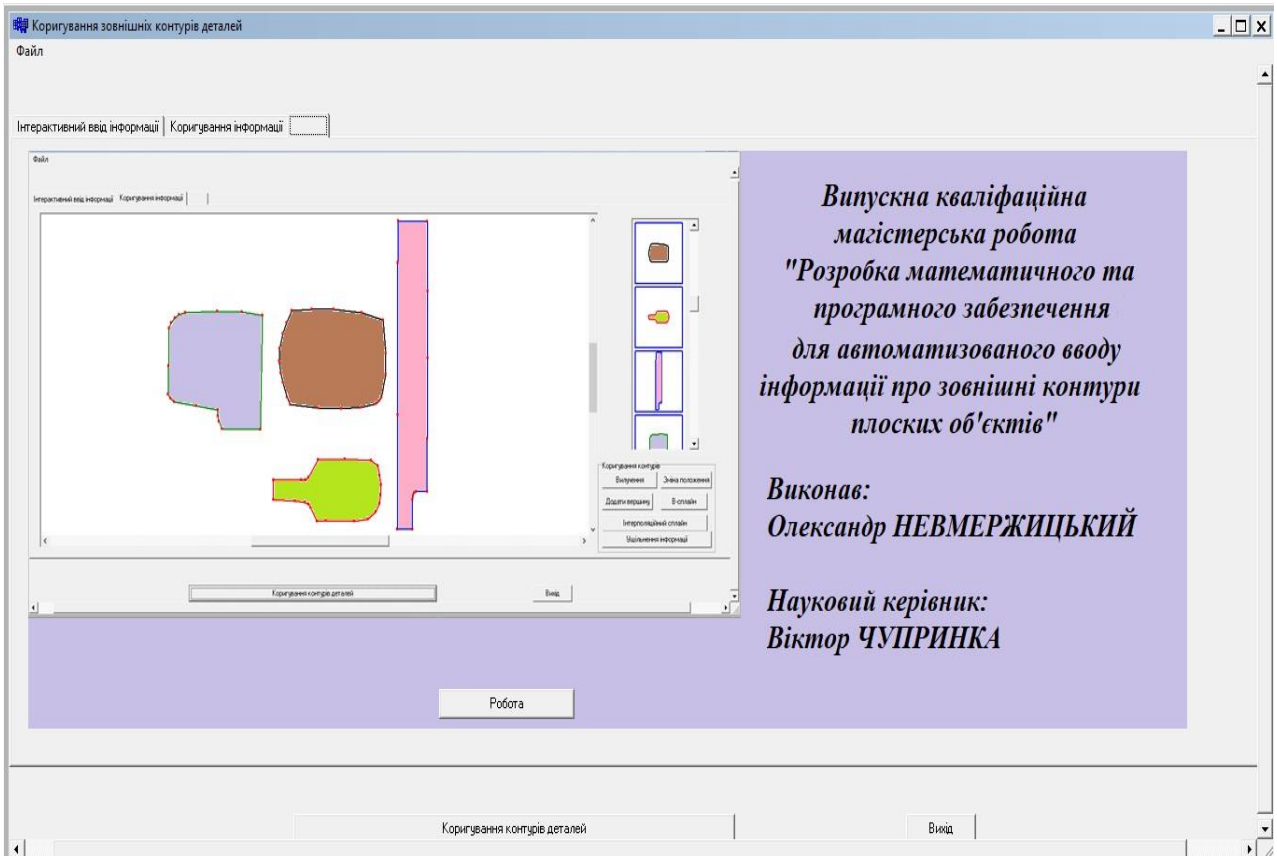


Рис. 3.1

Після натиску на кнопку «Робота» головне вікно програми наступний вигляд (рис. 3.2).

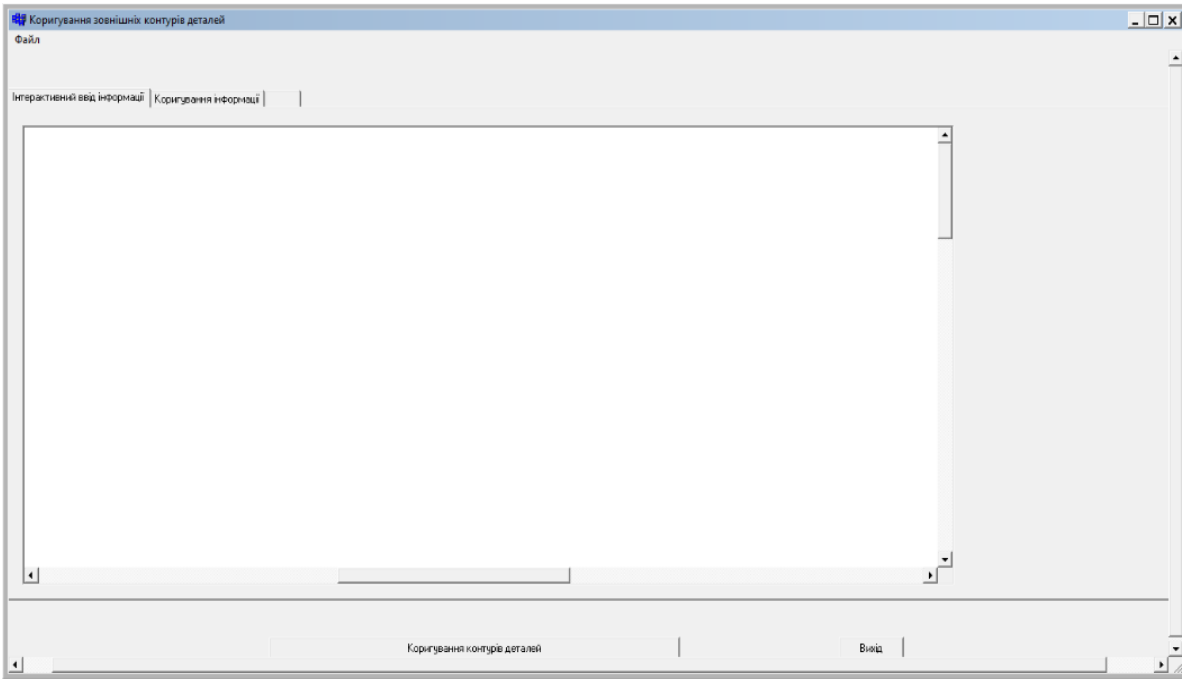


Рис. 3.2

Після вводимо файл *.bmp з кресленням необхідних деталей моделі. Головне вікно програми прийме настуний вигляд (рис. 3.3).

.

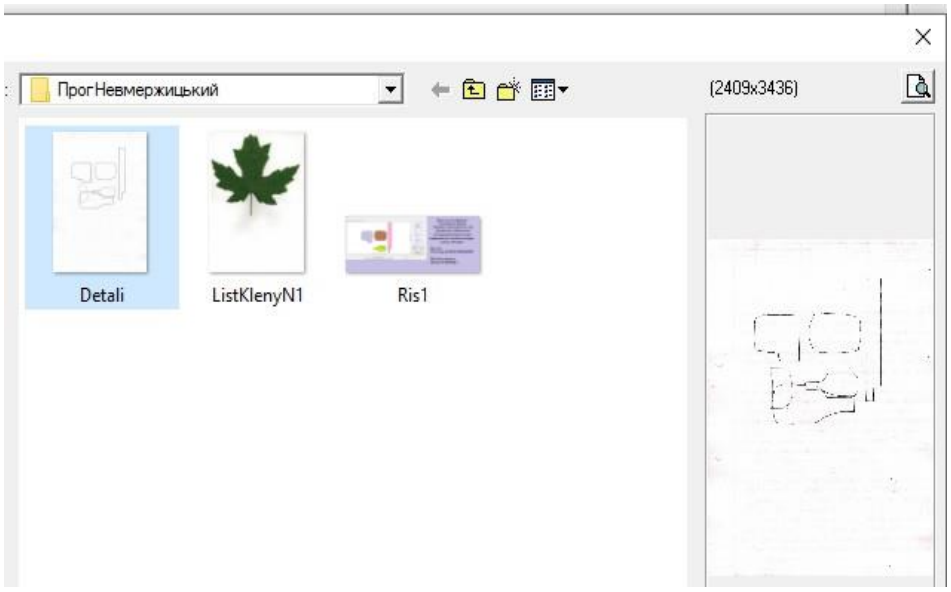


Рис. 3.3

Після введення файлу *.bmp з кресленням необхідних деталей моделі головне вікно програми прийме наступний вигляд (рис. 3.4).

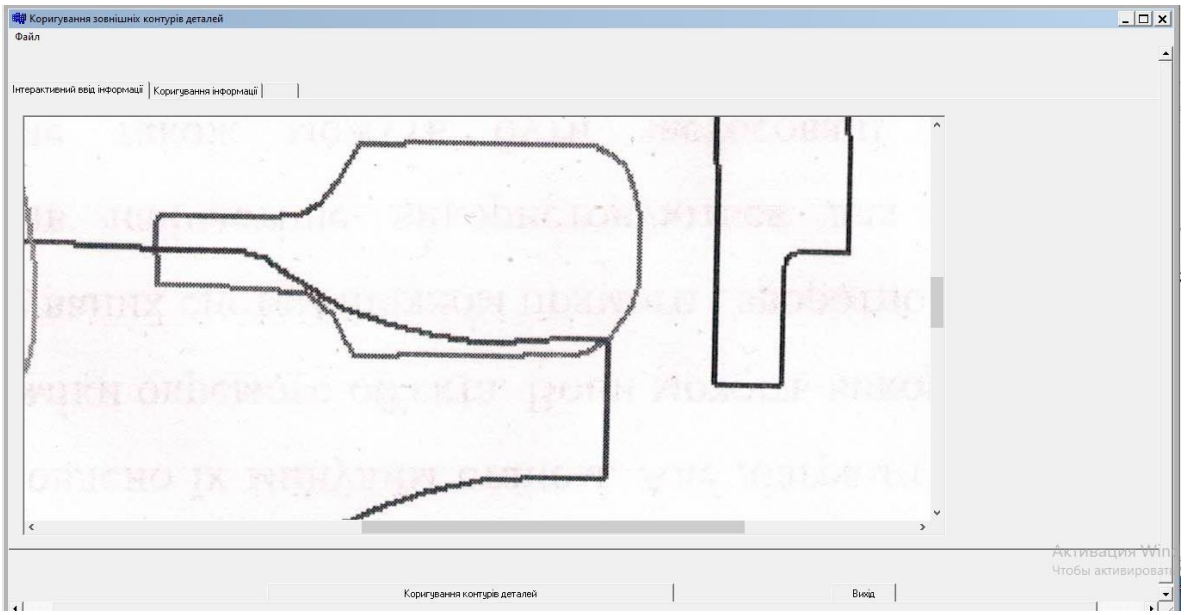


Рис. 3.4

Після інтерактивного введення інформації про зовнішні контури необхідних деталей, креслення яких представлене у файлі *.bmp, та натиску на кнопку «Коригування контурів деталей» головне вікно програми настуний вигляд (рис. 3.5).

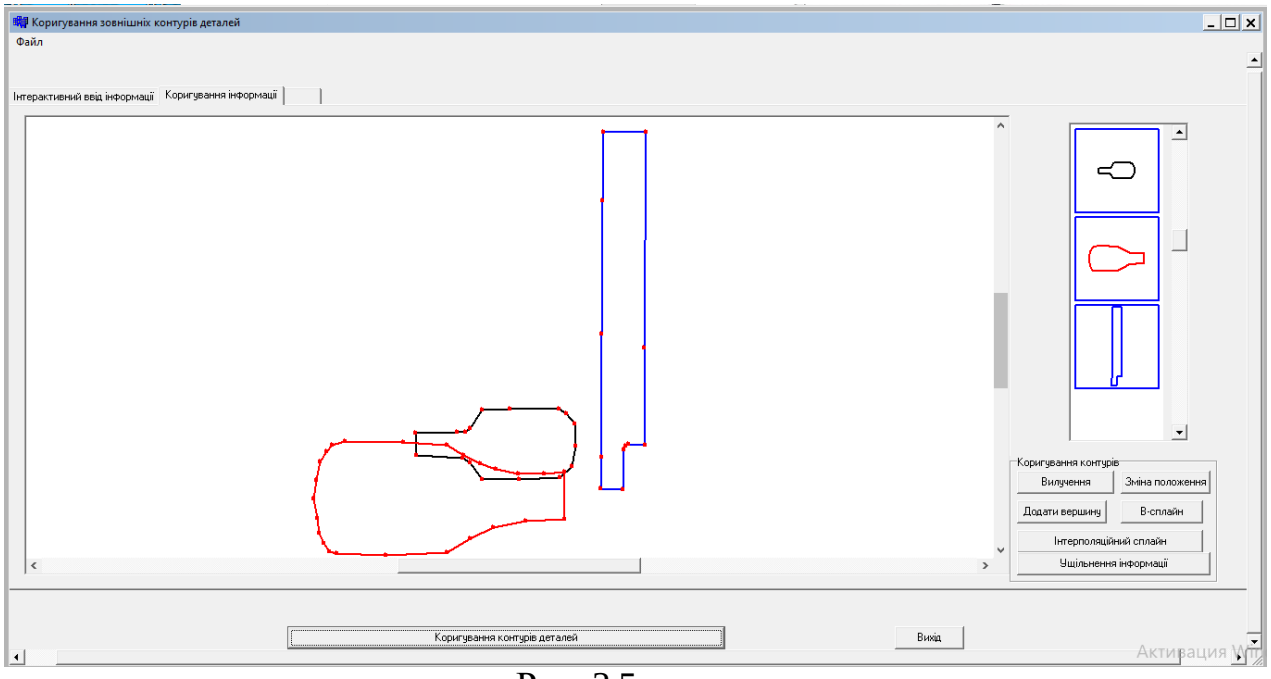


Рис. 3.5

В програмі є можливість відкоригувати зовнішні контури будь- якої із деталей, для якої введена інформація про зовнішні контури[21-30].

В програмному продукті передбачені наступні режими коригування:

- вилучення будь-якої вершини на зовнішньому контурі активної деталі;
- зніна положення будь-якої вершини на зовнішньому контурі активної деталі;
- додавання нової вершини на будь-якій стороні на зовнішньому контурі активної деталі;
- згладжування будь-якої ділянки на зовнішньому контурі деталі за допомогою В- сплайну;
- згладжування будь-якої ділянки на зовнішньому контурі деталі за допомогою інтерполяційного сплайну;
- ущільнення інформації про зовнішні контури деталей, тобто вилучення деяких вершин в апроксимуючому многокутнику не зменшуючи допустиму точність апроксимації.

Для цього необхідно вибрати потрібний режим коригування натиснутивши на відповідну кнопку в меню, що знаходиться в нижньому правому вікні основного вікна.

Крім того є можливість зберегти інформацію про зовнішні контури деталей у файлі *.dgt, інформацію про які ввели в інтерактивному режимі та відповідно відкоригували.

При необхідності цей файл *.dgt можна загрузити в програму та продовжити коригування контурів деталей, що були введені раніше в інтерактивному режимі.

Файл *.dgt із інтерактивно введеною інформацією про деталі має наступний вигляд:

Model – *імя моделі;*

AAAAA – *службова інформація;*

4 – *кількість деталей у моделі;*

Detal1– *назва 1-ої деталі;*

Detal2 – *назва 2-ої деталі;*

Detal3 – *назва 3-ої деталі;*

Detal4 – *назва 4-ої деталі;*

22 – *кількість вершин на зовнішньому контурі 1-ої деталі;*

23 – *кількість вершин на зовнішньому контурі 2-ої деталі;*

15 – *кількість вершин на зовнішньому контурі 3-ої деталі;*

19 – *кількість вершин на зовнішньому контурі 4-ої деталі;*

96.5202 -64.4315 Detal1
105.326 -63.9235
114.978 -63.9235
127.339 -65.1935
136.906 -67.7335
138.007 -78.9942
138.007 -86.1908
136.144 -94.2342
135.044 -95.6735
133.604 -96.4355
127.254 -97.5362
118.28 -98.0442
108.035 *-97.9595*
95.7582 -96.7742
94.4035 -94.0648
92.5408 -89.0695
91.4402 -85.8522
90.9322 -81.7882
91.1862 -77.4702
92.4562 -72.3055
94.0648 -68.4955
96.5202 -64.4315

Відповідно координати вершин X та У на зовнішньому контурі 1-ої деталі

88.2228 -122.682 Detal2
100.5 -122.852
102.447 -122.682
103.971 -121.751
108.12 -115.655
119.804 -115.486
131.403 -115.909
134.451 -117.941
135.213 -121.158
135.89 -124.291
135.806 -127.847
135.213 -130.81
……
104.733 -132.25
103.802 -131.149
102.193 -130.472
97.8749 -130.048
88.3922 -129.456
88.2228 -122.682

Відповідно координати вершин X та У на зовнішньому контурі 2-ої деталі

143.68 -33.5281 Detal3
156.464 -33.7821
156.549 -57.6581
156.634 -80.7722
156.38 -108.543
156.464 -126.662
151.469 -126.577
150.453 -127.085
150.03 -128.016
149.86 -129.371
149.776 -139.7
143.087 -139.616
143.256 -100.161
143.341 -47.8368
143.68 -33.5281

Відповідно координати вершин X та У на зовнішньому контурі 3-ої деталі

65.5321	-105.156	Detal4
82.3808	-105.41	
83.4815	-66.0401	
74.4221	-64.9395	
63.5001	-64.7701	
49.3608	-65.1935	
46.5668	-65.8708	
44.6194	-67.0561	
42.9261	-68.6648	
42.1641	-70.5275	
41.9101	-83.3968	
41.5714	-93.1335	
42.7568	-94.7422	
44.2808	-95.7582	
54.1021	-97.2822	
63.5848	-98.4675	
63.7541	-100.669	
64.1775	-102.532	
65.5321	-105.156	

Відповідно координати вершин X та Y на зовнішньому контурі 4-ої деталі

Висновки до третього розділу

1. Запропоновані алгоритми у другому розділі були реалізовані в програмний продукт для для автоматизованого введення та інтерактивного коригування інформації про зовнішні контури плоских геометричних обєктів

2. Цей програмний продукт легкий в користуванні, має дружній та привабливий інтерфейс. Може бути застосована на підприємствах легкої промисловості.

ВИСНОВКИ

Проведено дослідження та розроблені такі алгоритми для автоматизованого введення та інтерактивного коригування інформації про зовнішні контури плоских геометричних обєктів:

- автоматизованого введення інформації про зовнішні контури плоских геометричних обєктів;
- інтерактивного коригування інформації про зовнішні контури плоских геометричних обєктів:

  вилучення будь-якої вершини на зовнішньому контурі деталі;

  зміна положення будь-якої вершини на зовнішньому контурі деталі;

  додавання нової вершини на будь-якій стороні зовнішнього контуру деталі;

  виділення ділянки на зовнішньому контурі деталі;

  згладжування виділеної ділянки на зовнішньому контурі деталі за допомогою параметричного B-сплайну та параметричного інтерполяуійного сплайну.
- ущільнення інформації про зовнішні контури деталей.

Запропоноване математичне забезпечення для автоматизованого введення інформації про зовнішні контури плоских геометричних обєктів реалізоване в прогрограмний продукт. Цей програмний продукт легкий в користуванні, має дружній та привабливий інтерфейс. Може бути застосована на підприємствах легкої промисловості.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Горобець С.М., 2006. Основи комп'ютерної графіки. Київ: Центр навчальної літератури. Manual of Shoemaking. /Under edition of R.G. Miller.Produced By the Trading Department Clarks, 1989. - 337 p.
2. . Михайленко В. Є. Інженерна та комп'ютерна графіка. К. : Вища школа, 2002. 332 с.
3. Чупринка В.І. Підготовка інформації для автоматичного розкрою / В.І. Чупринка, Волошин, Піпа Т.А. // Вісник ДАЛПУ. – 2000. - №1. – С. 91- 83.
4. Чупринка В.І. Підготовка інформації для побудови раціональних схем розкрою рулонних матеріалів на деталі взуття та графічна візуалізація цих схем / В.І. Чупринка, О.О. Шишкіна, Л.Т. Свістунова // Вісник Технологічного університету Поділля. – 2003. - №5. - Ч1. – С. 38-41
5. Михайленко В. Є. Інженерна та комп'ютерна графіка. К. : Вища школа, 2002. 332 с.
6. Чупринка В.І. Система підготовки інформації для автоматичного розкрою спроектованих схем розкрою./ В.І. Чупринка, В.Ю. Щербань, І.І. Тонн // Вісник КНУТД. – 2003. - №2 . – С. 171-173.
7. Омельченко П.В. Автоматизована підготовка інформації про контури деталей шкіргалантерейних виробів / П.В. Омельченко, В.П. Коновал, В.І. Чупринка // Вісник КНУТД. – 2004. - № 4. – С. 138-142 Є. П. Зайцев: Вища математика. Лінійна та векторна алгебра, аналітична геометрія, вступ до математичного аналізу Алерта 574 с.Чупринка Н. В. Програмне забезпечення для генерування декоративних елементів на деталях жіночих сумок / Н. В. Чупринка // Тези доповідей XIV Всеукраїнської наукової конференції молодих вчених та студентів, 23-24 квітня 2015 р. „Наукові розробки молоді на сучасному етапі". – Т1 - К.:КНУТД, 2015. – С. 72.

8. Чупринка В.І. Алгоритм підготовки інформації для побудови розкрійних схем рулонних матеріалів на деталі взуття та шкіргалантерейних виробів / В.І. Чупринка, О.З Колиско // Вісник КНУТД. – 2005. - №3 – С. 19-24.
9. Чупринка В.І. Програмні методи підготовки інформації для автоматизованого розкрою матеріалу прямокутної форми / В.І. Чупринка, О.О. Хоменко, М.М. Шкоденко // Проблеми програмування. – 2008. -№2-3. - С. 665-668
10. Рудавський Ю. К. Лінійна алгебра та аналітична геометрія [Текст]: навч. посібн. / Ю. К. Рудавський, П. П. Костробій, Х. П. Луник, Д. В. Уханська, ДУ «Львівська політехніка», 1999. — 262 с.
11. Аналітична геометрія Підручник для вищих навчальних закладів Б. В. Гринев, І. К . Кириченко Гімназія 344 с.
12. .Караванова Т. П. Основи алгоритмізації та програмування. Форум. – К.: 2002. – 286 с.
13. .Ковалюк Т. В. Алгоритмізація та програмування: підручник з грифом МОН України / Т.В. Ковалюк. – Львів : Магнолія-2006, 2013. – 400 с.
14. Шаховська, Н. Б. Алгоритми і структури даних [Текст]: посібник / Н.Б. Шаховська, Р.О. Голощук; - Львів: Магнолія, 2010 – 215с.
15. Веселовська Г.В. та Ходакова, В.Є., 2015. Комп’ютерна графіка. Київ: Кондор.
16. . Михайленко В. Є. Інженерна та комп’ютерна графіка. К. : Вища школа, 2002. 332 с.
17. Горобець С.М., 2006. Основи комп’ютерної графіки. Київ: Центр навчальної літератури. Manual of Shoemaking. /Under edition of R.G. Miller.Produced By the Trading Department Clarks, 1989. - 337 p.
18. Березовський В.С., Потієнко, В.О. та Завадський, І.О., 2009. Основи комп’ютерної графіки. Київ: BHV.
19. Ванін В.В., Перевертун, В.В. та Надкернична, Т.М., 2006. Комп’ютерна інженерна графіка в середовищі AutoCAD. Київ: Каравела. 2023, 384 с.

20. Романюк О.Н., 2001. Комп’ютерна графіка. Вінниця: Вінницький державний технічний університет.
21. Роберт Мартін Чиста архітектура. Мистецтво розроблення програмного забезпечення.Видавництво — Фабула Жанр, ISBN — 978-617-09-5286-Рік видання — 2019, 416 с,
22. Ерік Фрімен , Елізабет Робсон Патерни проєктування, Видавництво — Фабула Жанр, Видавництво — Фабула Жанр, ISBN — 978-617-09-6159-4, Рік видання — 2020, 672 с.
23. Роберт Мартін Чистий Agile, Видавництво — Фабула Жанр, ISBN — 978-617-09-6760-2 , Рік видання — 2021, 224 с.
24. Олексій Васильєв Програмування мовою PYTHON Видавництво — Навчальна книга Богдан Жанр, ISBN — 978-966-10-5611-3, 504 с, ,
25. Берт Бейтс , Кеті Сьєрра JAVA, Видавництво — Фабула Жанр, ISBN — 978-617-522-033-7, Рік видання — 2022, 720 с.
26. Пол Беррі Head First Python, Видавництво — Фабула Жанр, ISBN — 978-617-522-019-1 , Рік видання — 2021 ,624 с.
27. Дж. Генк Рейнвотер Як пасти котів. Посібник для програмістів, які мають керувати іншими програмістами, Видавництво — Фабула Жанр, ISBN — 978-617-09-6155-6, Рік видання — 2020, 320 с.
28. Томас Г. Кормен, Чарлз Е. Лейзерсон, Роналд Л. Рівест, Кліфорд Стайн Вступ до алгоритмів, Видавництво : К.І.С., ISBN : 9786176842392,Рік видання : 2019, 1288 с.
29. Роберт Бош Opt Art. Від математичної оптимізації до візуального дизайну Видавництво : Фабула, ISBN : 9786175220795, 200 с.10.
30. Джин Кім, Джез Хамбл, Патрік Дебуа, Джон Вілліс DevOps. Посібник. Як домогтися гнучкості, надійності і безпеки світового рівня в технічних компаніях Видавництво : Фабула, ISBN : 9786170979841, Рік видання : 72. Зайченко Ю.П. Дослідження операцій: Підручник. – К.: ВІПОЛ, 2010.

Додаток А

Публікації по темі випускної кваліфікаційної магістерської роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Київський національний університет технологій та дизайну

Кафедра комп'ютерних наук

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В НАУЦІ, ВИРОБНИЦТВІ ТА ПІДПРИЄМНИЦТВІ

Збірник наукових праць молодих вчених, аспірантів, магістрів кафедри комп'ютерних наук

Київ – 2023

Якщо $|[OA_{j+1} \times OA_{j+1}]| > 0$, то кут буде додатнім, якщо $|[OA_{j+1} \times OA_{j+1}]| < 0$, то кут буде від'ємним, якщо $|[OA_{j+1} \times OA_{j+1}]| = 0$, то кут дорівнює нулю.

Так як не завжди вдається описати зовнішні контури плоских геометричних об'єктів аналітично, то ми апроксимуємо їх многокутниками. Будемо вважати, що два многокутника не перетинаються, якщо жодна вершина одного многокутника не знаходиться всередині іншого многокутника. Для визначення цього і використаємо метод кутів

Висновки

Запропоноване математичне та програмне забезпечення для інтерактивної побудови щільних укладок плоских геометричних об'єктів має практичну значимість та може бути використане для визначення економічності моделей у взуттєвому виробництві.

НЕВМЕРЖИЦЬКИЙ О.А., ЧУПРИНКА В.І.

РОЗРОБКА МАТЕМАТИЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗОВАНОГО ВВОДУ ІНФОРМАЦІЇ ПРО ЗОВНІШНІ КОНТУРИ ПЛОСКИХ ОБ'ЄКТІВ

NEVMERZHYTSKY O.A., CHUPRYNKA V.I.

DEVELOPMENT OF MATHEMATICAL AND SOFTWARE FOR AUTOMATED INPUT OF INFORMATION ABOUT OUTER CONTOURS OF FLAT OBJECTS

The article is devoted to the development of mathematical and software for the automated input of information about the external contours of flat objects. The software has a user-friendly interface and does not require additional computer science knowledge when working with it.

Key words: external contour, interactive preparation of information, flat geometric object

Вступ

В умовах ринкової економіки необхідно створення науково обґрунтованих методів проектування раціональних схем розкрою матеріалів та автоматизація самого процесу проектування з ціллю отримання об'єктивних результатів за короткий період часу.

Раціональні й економічні витрати матеріальних і енергетичних ресурсів, а також захист навколишнього середовища від забруднення були завжди і є пріоритетними напрямками в розвитку України. А для цього необхідно зменшувати кількість відходів. Так матеріали складають понад 80% у собівартості взуття, а технологічні особливості виробництва взуття призводять до того, що тільки відходи розкрою взуттєвих матеріалів складають понад 20%, тому очевидна актуальність раціонального використання матеріалів. А отримання інформації про зовнішні контури

плоских геометричних об'єктів є першим етапом задачі раціонального розкрою.

Основна частина

Контури деталей виробів легкої промисловості в більшості випадків мають складну конфігурацію, тому їх апроксимують більш простими кривими. Найбільше розповсюдження одержала кусково-лінійна апроксимація, тобто апроксимація кривими першого порядку (прямими). Це зручно також для автоматизованого вводу такої інформації при використанні спеціальних пристроїв.

Нехай S - фігура з заданою орієнтацією. Зв'яжемо з деталлю S координатну систему XOY, де O – полюс деталі, обраний в будь-якій її внутрішній точці. Контур взуттєвої деталі S апроксимуємо ламаною лінією з вершинами в послідовно вибраних на контурі деталі точках (рис. 1.).

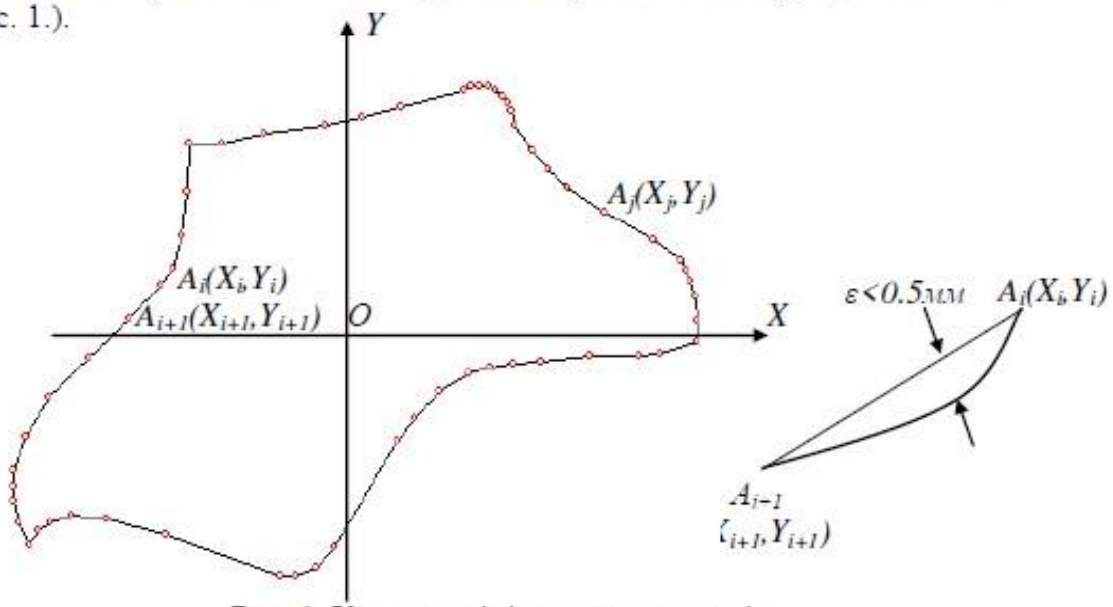


Рис. 1. Кусково-лінійна апроксимація

Будь-яка деталь S може бути представлена координатами вершин опукло-ввігнутого многокутника, тобто масивом $\{X_j, Y_j\}$,. $j=1..n$, а координати будь-якої точки на прямій між вершинами A_j та A_{j+1} можна визначити наступним чином

$$X = X_j + (X_{j+1} - X_j)t$$

$$Y = Y_j + (Y_{j+1} - Y_j)t \qquad (1)$$

$$0 \le t \le 1;\ j = 1,2..n$$

де j– порядковий номер відрізка між вершинами A_j та A_{j+1} ;
n– кількість точок апроксимації для деталі S;

t– параметр, що визначає положення точки на прямій між вершинами A_j та A_{j+1}.

Постановка задачі. Нехай маємо файл формату **.bmp* із кресленнями деталей виробів легкої промисловості. Необхідно отримати інформацію про зовнішні контури цих деталей у вигляді координат апроксимуючого многокутника.

В будь-якому файлі формату **.bmp* в перших двох (0-1) байтах зберігаються символи 4D42h, які є буквами 'BM'; в наступних чотирьох (2-5) байтах зберігається розмір файлу в байтах; наступні чотири(6-9) байти – резервні поля; наступні чотири(10-13) байти – зсув, з якого починається саме зображення (растр); наступні чотири (14-17) байти – розмір заголовка BITMAP у байтах (дорівнює 40); наступні чотири (18-21) байти - ширина зображення в пікселах; наступні чотири (22-25) байти - висота зображення в пікселах; наступні два (26-27) байти - кількість площин, повинно бути 1; наступні два (28-29) байти – кількість біт на відображення одного піксела: 1, 4, 8 або 24; наступні чотири (30-33) байти - тип стиску; наступні чотири (34-37) байти - 0 або розмір стиснутого зображення в байтах; наступні чотири (38-41) байти - горизонтальна роздільна здатність, *піксел/м*; наступні чотири (42-47) байти - вертикальна роздільна здатність, *піксел/м*; наступні чотири (46-49) байти - кількість використовуваних кольорів; наступні чотири(50-53) байти - кількість "важливих" кольорів. Далі йде інформація про пікселі самого зображення по рядкам.

Необхідно зауважити, що якщо інформація записана в декількох байтах, то перший байт є наймолодшим. Для виводу креслення на екран монітору нас будуть цікавити 18-21 байти - ширина зображення в пікселах та 22-25 байти - висота зображення в пікселах. Для визначення реальних координат вершин апроксимуючих многокутників для зовнішніх контурів деталей взуття, креслення яких представлені у файлі формату **.bmp*, нас будуть цікавити 38-41 байти - горизонтальна роздільна здатність в *піксел/мм* та 42-47 байти - вертикальна роздільна здатність в *піксел/мм*.

Нехай нам потрібно визначити ширину зображення в пікселах *ShIm*, що зберігається в 18-21 байтах, та значення 18, 19, 20, 21 байтів, що відповідно позначимо: A, B, C, D.

Тоді *ShIm* можна визначити за наступною формулою:

$$ShIm = A + 256 * B + 256^2 * C + 256^3 * D \qquad (2)$$

Аналогічно визначається висота зображення, горизонтальна та вертикальна роздільна здатність в *піксел/м*.

Нехай координати вершин апроксимуючого многокутника для зовнішнього контуру деталі взуття в пікселах представлено

масивом $\{X_j^E, Y_j^E\}$, $j=1..n$. Тоді реальні координати вершин в *мм* можна розрахувати за наступними формулами:

$$\begin{aligned} X_j &= mx \cdot X_j^E \\ Y_j &= my \cdot Y_j^E \end{aligned}, \quad \text{де} \quad \begin{aligned} mx &= 1000/RSh \\ my &= 1000/RVs \end{aligned}, \quad j = 1,2,...n \qquad (3),$$

де RSh- горизонтальна роздільна здатність в *піксел/м*,

RVs - вертикальна роздільна здатність в *піксел/м*.

На основі формул (1-3) розроблений алгоритм інтерактивної підготовки інформації про зовнішні контури плоских геометричних об'єктів, який реалізований в програмний продукт інтерактивної підготовки інформації про зовнішні контури плоских геометричних об'єктів.

Висновки

Запропоноване математичне та програмне забезпечення для продукт інтерактивної підготовки інформації про зовнішні контури плоских геометричних об'єктів має практичну значимість, так як воно направлене на підвищення конкурентоспроможності вітчизняного малого виробництва.

ОВЧАРОВ А.С., ЧУПРИНКА В.І.

РОЗРОБКА МАТЕМАТИЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРОЄКТУВАННЯ ЧОЛОВІЧИХ ШТАНІВ НА НЕ ТИПОВУ ФІГУРУ

OVCHAROV A.S., CHUPRINKA N.V.

DEVELOPMENT OF MATHEMATICAL AND SOFTWARE FOR DESIGNING MENTROUSERS FOR AN Atypical FIGURE

The article is devoted to the development of mathematical and software for designing men's pants for non-typical figures. The software has a user-friendly interface and does not require additional computer science knowledge when working with it.

Key words: software, men's pants, non-typical figure information preparation, flat geometric object

Вступ

Часта зміна моделей одягу потребує значного підвищення підготовчих робіт. Скорочення термінів цих робіт, зменшення вартості та підвищення якості технологічних рішень повинно бути досягнуто шляхом підвищення продуктивності за рахунок впровадження у виробництво математичних методів, обчислювальної техніки та розробки програмних засобів технологічної підготовки виробництва. Це зумовлює необхідність створювати у легкій промисловості гнучкі виробничі системи, які швидко і з мінімальними затратами могли переналагоджуватись на випуск нової продукції.

Міністерство освіти і науки України

Київський національний університет
технологій та дизайну

МЕХАТРОННІ СИСТЕМИ: ІННОВАЦІЇ ТА ІНЖИНІРИНГ

ТЕЗИ ДОПОВІДЕЙ VII МІЖНАРОДНОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ

23 листопада 2023

Рекомендовано Вченою радою
факультету мехатроніки та комп'ютерних технологій
Київського національного університету технологій та дизайну

КИЇВ 2023

УДК 688.359

РОЗРОБКА МАТЕМАТИЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗОВАНОЇ ПІДГОТОВКИ ІНФОРМАЦІЇ ПРО ДЕТАЛІ ВИРОБІВ ЛЕГКОЇ ПРОМИСЛОВОСТІ

Н.В. Чупринка, кандидат технічних наук
Київський національний університет технологій та дизайну
О.А. Невмержицький, магістрант
Київський національний університет технологій та дизайну
М.С. Каземирчик, магістрант
Київський національний університет технологій та дизайну

Ключові слова: програмне забезпечення, вироби легкої промисловості, автоматизоване проектування.

Для підвищення рівня розвитку нашої легкої промисловості та її конкурентоздатності, необхідна автоматизація та програмна підтримка при розробці проектів в легкій промисловості. Метою автоматизації проектування виробів текстильної та легкої промисловості є підвищення якості продукції, зниження матеріальних витрат на виготовлення, скорочення термінів проектування. Це можна досягти впровадженням сучасних технологій із застосуванням інформаційних технологій у виробництво.

Так як деталі виробів виробів легкої промисловості мають складну конфігурацію і здебільше їх контури не можна описати аналітично, то ми їх контури будемо апроксимувати. Найбільшого розповсюдження отримала кусково-лінійна апроксимація.

Кусково-лінійна апроксимація являє собою заміну дійсного контуру деталі відрізками прямих, причому відстань від відрізків до точок контуру не перевищує похибки апроксимації. Найбільше розповсюдження отримав кусково-лінійний спосіб апроксимації. При цьому способі апроксимації зовнішній контур деталі апроксимується багатокутником. Довжина сторін багатокутника залежить від кривизни контуру і взятої погрішності апроксимації.

При кусково-лінійній апроксимації деталі виробів шкіргалантереї представляють собою многокутники. Многокутник однозначно описується координатами його вершин та послідовністю цих вершин. Тоді многокутник A однозначно буде визначатися масивом з координатами його вершин, тобто $A\{Xa_i, Ya_i\}, i=1,2..n$.

Часто при підготовці інформації про зовнішні контури деталей необхідно коригування цих контурів. В роботі запропоновані наступні режими коригування:

- вилучення будь-якої вершини на зовнішньому контурі деталі;
- зміна положення будь-якої вершини на зовнішньому контурі деталі;
- додавання нової вершини на зовнішньому контурі деталі;

- згладжування будь-якої ділянки на зовнішньому контурі деталі за допомогою параметричного сплайну.

Зупинимося більш детально на режимі згладжування будь-якої ділянки на зовнішньому контурі деталі за допомогою параметричного сплайну, так як він є найбільш складним.

Для згладжування застосуємо параметричний В-сплайн.

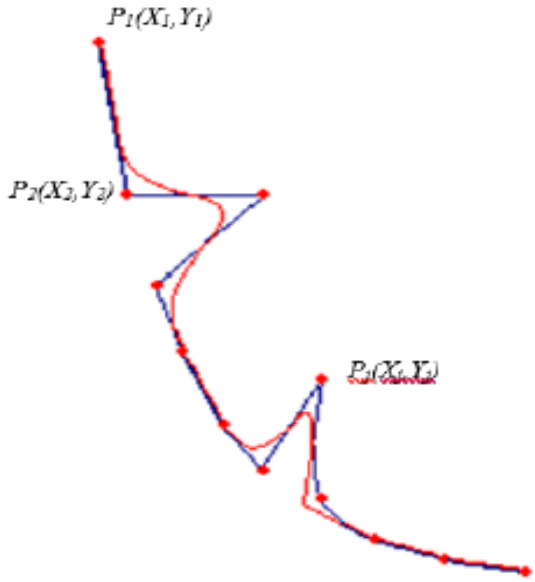


Рисунок 1 - Згладжування ділянки контуру деталі за допомогою параметричного В-сплайну

Для того, щоб провести згладжування відповідної ділянки зовнішнього контуру деталі потрібно:

- вказати крайні точки ділянки зовнішнього контуру деталі, які необхідно згладити;
- продублювати ці точки;
- ідентифікувати цю ділянку зовнішнього контуру деталі;
- згладити вибрану ділянку на контурі деталі за допомогою параметричного В-сплайну;
- перенумерувати вершини апроксимуючого багатокутника таким чином, щоб ділянка контуру, що необхідно згладити, знаходилась в кінці;
- провести згладжування;
- замінити ділянку контуру, що необхідно згладити, новим контуром.

Висновки

Запропоноване математичне та програмне забезпечення для автоматизованої підготовки інформації про деталі виробів легкої промисловості має практичну значимість, так як воно направлене на підвищення конкурентоспроможності вітчизняного малого виробництва.

Листинг програмного продукту

```
//---------------------------------------------------

#include <vcl.h>
#include <math.h>
#include <Math.hpp>
#include <fstream.h>
#include <iostream.h>
#include <conio.h>


//---------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

#include <vcl.h>
#pragma hdrstop

#include "UnNewmer.h"
//---------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------
AnsiString Model, NameDet[40];
int KilksPointDet[40];
int KilDet,Xt[300],Yt[300],N0;
float
Xd[40][300],Yd[40][300],Xs[100],Ys[100],XcDet[40],YcDet[40],DlDet[40],ShDet[4
0],
    DlMod,ShMod,XcMod,YcMod,Xq,Yq,X0,Y0,X1,Y1,X2,Y2,X3, X4, Y3,
Y4,Xh,Yh,DM,ShVitZ,ShVitV,HVit;
 ifstream f;
int i, p, NDet,NPointD;
float mx, my, mxyIm2, mxyIm3;
bool
Fl_Btn3,Fl_Btn4,Fl_Btn5,Fl_Btn6,Fl_Btn7,Fl_Btn10,Fl_Btn13,Fl_Btn14,Fl_Btn11,F
l_Cor,Fl_Btn16,Fl_Btn15;
int NLeft,NRight,Nb,Ne,Nr1,Nr2,Nr,R0;
```

```
bool Fl_Left,Fl_Right;
float Xr1[300],Yr1[300],Xr2[300],Yr2[300],Xr[300],Yr[300],R;
AnsiString Str;
float Xo,Yo,Xa1,Ya1,Xc1,Yc1,XoN,YoN,Xa1N,Ya1N,Xc1N,Yc1N;
float MyAngle(float X, float Y,float Xo, float Yo);
int xLeft, yLeft, xRight, yRight;
float xLeftF, yLeftF, xRightF, yRightF;
float XX[10], YY[10];
int NN[10], jj = 0;
bool Fl;
float MaxX[40],MaxY[40],MinX[40],MinY[40],MaxDl,MaxSh;

  void ChangePoint(int n, float X[],float Y[],int NP, float XNew,float YNew);

//-----------------------------------------
 int Val(int k)
  { int Q;
    unsigned int A,B,C,D;
    f.seekg(k);
    A=0;B=0;C=0;D=0;
    f.read((char*)&A,1);
    f.read((char*)&B,1);
    f.read((char*)&C,1);
    f.read((char*)&D,1);
    Q=A+256*B+256*256*C+256*256*256*D;
    return Q;
   }
//-----------------------------------------
void __fastcall TForm1::Bmp1Click(TObject *Sender)
{
  unsigned int A,B,C,D;
int i,L,DlIm, ShIm,RzDl,RzSh;
float Xr[100],Yr[100];
char  Fname[100],T;
AnsiString NameF;

if (OpenPictureDialog1->Execute())
  Image1->Picture->LoadFromFile(OpenPictureDialog1->FileName);
  NameF=OpenPictureDialog1->FileName;
   for (i=0;i<99;i++)Fname[i]=NULL;
//   ifstream f;
   L=NameF.Length();
```

```
  for(i=1; i<=L;i++)
  {
   T=NameF[i];
   Fname[i-1]=T;
  }
  f.open(Fname,ios::in);
  DlIm=Val(18);
  ShIm=Val(22);
  RzDl=Val(38);
  RzSh=Val(42);
  f.close();
  mx=1000./RzDl;
  my=1000./RzSh;
  Image1->Width=DlIm;
  Image1->Height=ShIm;
}


//---------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
i=0; p=0;
 Fl_Btn3=false;
 Fl_Btn4=false;
 Fl_Btn5=false;
 Fl_Btn6=false;
 Fl_Btn7=false;
 Fl_Btn10=false;
 Fl_Btn11=false;
 Fl_Cor=false;
 Fl_Right=false;
 Fl_Left=false;
 PageControl1->ActivePage=TabSheet3;

}
//---------------------------------------------------
void __fastcall TForm1::Image1MouseDown(TObject *Sender,
     TMouseButton Button, TShiftState Shift, int X, int Y)
{
  int Xr,Yr;
  Image1->Canvas->Pen->Mode=pmXor;
  Image1->Canvas->Pen->Color=clGreen;
```

```
  Image1->Canvas->Pen->Width=2;
  if (Button==mbLeft)
   {
    Xd[p][i]=X*mx; Xt[i]=X;
    Yd[p][i]=-Y*my; Yt[i]=Y;
    if (i==0)Image1->Canvas->MoveTo(Xt[i],Yt[i]);
    else Image1->Canvas->LineTo(Xt[i],Yt[i]);
    i++;
   }
  else
   if (Button==mbRight)
    {
     if (i>0)
      {
       i--;
       Image1->Canvas->LineTo(Xt[i],Yt[i]);
      }
    }
else
    {
     // i++;
      Xd[p][i]=Xd[p][0];
      Yd[p][i]=Yd[p][0];
      Image1->Canvas->LineTo(Xt[0],Yt[0]);
      p=p+1;
      if (p==1)  Model="AAAAA";
      NameDet[p-1]=IntToStr(p);
      KilksPointDet[p-1]=i+1;
      KilDet=p;
      i=0;
      }
}
//---------------------------------------

//-------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
  Form1->Close();
}
//-----------------------
float MaxMin(int n,float Z[],int p)
     {
```

```
      int i;
      float q;
      q=Z[0];
      for (i=1;i<n;i++)
       if (p*q<p*Z[i]) q=Z[i];
      return q;
     }
//-------------------------------------------------
 void ParamDet(int ND)
  {
    MaxX[ND]=MaxMin(KilksPointDet[ND],Xd[ND],1);
    MaxY[ND]=MaxMin(KilksPointDet[ND],Yd[ND],1);
    MinX[ND]=MaxMin(KilksPointDet[ND],Xd[ND],-1);
    MinY[ND]=MaxMin(KilksPointDet[ND],Yd[ND],-1);
    DlDet[ND]=MaxX[ND]-MinX[ND];
    ShDet[ND]=MaxY[ND]-MinY[ND];
    XcDet[ND]=(MaxX[ND]+MinX[ND])/2;
    YcDet[ND]=(MaxY[ND]+MinY[ND])/2;
   }
//=================================================
  void ParamModeli()
  {
  int i;
  float Xmax,Ymax,Xmin,Ymin;
   Xmax=MaxX[0]; Xmin=MinX[0];
   Ymax=MaxY[0]; Ymin=MinY[0];
   for(i=1;i<p;i++)
    {
    if(Xmax<MaxX[i]) Xmax=MaxX[i];
    if(Ymax<MaxY[i]) Ymax=MaxY[i];
    if(Xmin>MinX[i]) Xmin=MinX[i];
    if(Ymin>MinY[i]) Ymin=MinY[i];
    }
    DlMod=Xmax-Xmin;
    ShMod=Ymax-Ymin;
    XcMod=(Xmax+Xmin)/2;
    YcMod=(Ymax+Ymin)/2;
   }
//-------------------------------------------------
 void GraphIm3(int  n, float X[], float Y[], float Xcf, float Ycf,
              float Xce, float Yce, float mxy,int q, int p)
    {
```

```
 int j,Xr[300],Yr[300];
 for(j=0;j<n;j++)
  {
   Xr[j]=floor((X[j]-Xcf)*mxy+Xce);
   Yr[j]=floor(-(Y[j]-Ycf)*mxy+Yce);
  }
Form1->Image3->Canvas->Pen->Width=p;
Form1->Image3->Canvas->Pen->Mode=pmCopy;
switch(q)
  {
   case 1:Form1->Image3->Canvas->Pen->Color=clRed;break;
   case 2:Form1->Image3->Canvas->Pen->Color=clBlue;break;
   case 3:Form1->Image3->Canvas->Pen->Color=clGreen;break;
   case 4:Form1->Image3->Canvas->Pen->Color=clGray;break;
   default:Form1->Image3->Canvas->Pen->Color=clBlack;
  }
 for(j=0;j<n;j++)
  if(j==0)Form1->Image3->Canvas->MoveTo(Xr[j],Yr[j]);
  else Form1->Image3->Canvas->LineTo(Xr[j],Yr[j]);
}
//---------------------------------------------------
void GraphIm2(int n, float X[], float Y[], float Xcf, float Ycf,
        float Xce, float Yce, float mxy,int q, int p)
{
 int j,Xr[300],Yr[300];
 for(j=0;j<n;j++)
  {
   Xr[j]=floor((X[j]-Xcf)*mxy+Xce);
   Yr[j]=floor(-(Y[j]-Ycf)*mxy+Yce);
  }
Form1->Image2->Canvas->Pen->Width=p;
Form1->Image2->Canvas->Pen->Mode=pmCopy;
switch(q)
  {
   case 1:Form1->Image2->Canvas->Pen->Color=clRed;break;
   case 2:Form1->Image2->Canvas->Pen->Color=clBlue;break;
   case 3:Form1->Image2->Canvas->Pen->Color=clGreen;break;
   case 4:Form1->Image2->Canvas->Pen->Color=clGray;break;
   default:Form1->Image2->Canvas->Pen->Color=clBlack;
  }
 for(j=0;j<n;j++)
  if (j==0)Form1->Image2->Canvas->MoveTo(Xr[j],Yr[j]);
```

```
      else Form1->Image2->Canvas->LineTo(Xr[j],Yr[j]);
     }

//-----------------------------------------------

 void Elipse(float X, float Y, int r, float Xcf, float Ycf,
          float Xce, float Yce, float mxy)
 {
  int X0,Y0;
     X0=floor((X-Xcf)*mxy+Xce);
     Y0=floor(-(Y-Ycf)*mxy+Yce);
     Form1->Image2->Canvas->Pen->Color=clRed;
    // Form1->Image2->Canvas->Pen->Color=clBlack;
     Form1->Image2->Canvas->Pen->Width=2;
     Form1->Image2->Canvas->Ellipse(X0-r,Y0-r,X0+r,Y0+r);
 }
 //-----------------------------------------------
 void BuildIm2(float XcE,float YcE)
 {
  int i,j;
   mx=(Form1->ScrollBox2->Width-40)/DlMod;
   my=(Form1->ScrollBox2->Height-40)/ShMod;
   mxyIm2=mx;
   if (my<mx)mxyIm2=my;
   for(i=0;i<p;i++)
    {
     GraphIm2(KilksPointDet[i],Xd[i],Yd[i],XcMod,YcMod, XcE, YcE, mxyIm2, i,
2);
    for (j=0;j<KilksPointDet[i]; j++)
      Elipse(Xd[i][j],Yd[i][j],2,XcMod,YcMod, XcE, YcE, mxyIm2);
     }
   }
//=========================================
  void MaxDl_MaxSh()
 {
  int i;
   MaxDl=DlDet[0]; MaxSh=ShDet[0];
   for(i=1;i<p;i++)
    {
     if (MaxDl<DlDet[i])MaxDl=DlDet[i];
     if (MaxSh<ShDet[i])MaxSh=ShDet[i];
    }
```

```
   }

//-------------------------------------------------//----------------
void CleanIm3()
 {
  Form1->Image3->Canvas->Pen->Color=clWhite;
  Form1->Image3->Canvas->Brush->Color=clWhite;
  Form1->Image3->Canvas->Rectangle(0,0,Form1->Image3->Width,Form1-
>Image3->Height);
  }
//-------------------------------------------------//----------------
---------------------------------
void CleanIm2()
 {
  Form1->Image2->Canvas->Pen->Color=clWhite;
  Form1->Image2->Canvas->Brush->Color=clWhite;
  Form1->Image2->Canvas->Rectangle(0,0,Form1->Image2->Width,Form1-
>Image2->Height);
  }

//=================================================
void __fastcall TForm1::Image2MouseMove(TObject *Sender, TShiftState Shift,
     int X, int Y)
{

  float XcurF,YcurF,XcE,YcE;
  int j;
 if (Fl_Btn4&&Fl_Cor)
  if(Xq!=X||Yq!=Y)
   {
    Xq=X;
    Yq=Y;
    XcE=Image2->Width/2;
    YcE=Image2->Height/2;
    XcurF=(Xq-XcE)/mxyIm2+XcDet[NDet];
    YcurF=-(Yq-YcE)/mxyIm2+YcDet[NDet];
    ChangePoint(KilksPointDet[NDet],Xd[NDet],Yd[NDet],NPointD,XcurF,YcurF);
    CleanIm2();

GraphIm2(KilksPointDet[NDet],Xd[NDet],Yd[NDet],XcDet[NDet],YcDet[NDet],Xc
E,YcE,mxyIm2, NDet, 2);
    for (j=0;j<KilksPointDet[NDet]; j++)
```

```
Elipse(Xd[NDet][j],Yd[NDet][j],2,XcDet[NDet],YcDet[NDet],XcE,YcE,mxyIm2);
    }
  }

//---------------------------------------------------

//=====================================================
==========
  void GenRamkaIm3(int i,float& XcIm3,float& YcIm3)
   {
     float Xr[5],Yr[5];
       Xr[0]=5;  Yr[0]=5+95*i;
       Xr[1]=95; Yr[1]=5+95*i;
       Xr[2]=95; Yr[2]=95*(i+1);
       Xr[3]=5;  Yr[3]=95*(i+1);
       Xr[4]=5;  Yr[4]=5+95*i;
       XcIm3=50; YcIm3=95*i+50;
       GraphIm3(5, Xr, Yr, XcIm3, YcIm3, XcIm3, YcIm3, 1, 2, 2);
      }
//=====================================================
===============


  void BuildIm3(float& XcIm3,float& YcIm3)
 {
    MaxDl_MaxSh();
    mx=85/MaxDl; my=85/MaxSh;
    mxyIm3=mx;
    if (my<mx)mxyIm3=my;
    CleanIm3();
    for(i=0;i<p; i++)
     {
      GenRamkaIm3(i,XcIm3,YcIm3);
      GraphIm3(KilksPointDet[i],Xd[i],Yd[i],XcDet[i],YcDet[i], XcIm3, YcIm3,
            mxyIm3, i, 2);
      }
    }
//---------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  int i,j, ni;
```

```
 float XcE,YcE,XcIm3,YcIm3, q;

  XcE=Image2->Width/2;
  YcE=Image2->Height/2;
  PageControl1->ActivePage=TabSheet2;
 for(i=0;i<p;i++)
 {
  ParamDet(i); }
  ParamModeli();
  BuildIm2(XcE,YcE);
  BuildIm3(XcIm3,YcIm3);

}
//---------------------------------------------------
 void ReBuildIm3(float& XcIm3,float& YcIm3)
 {
  int i;
    for(i=0;i<p;i++) ParamDet(i);
    ParamModeli();
    MaxDl_MaxSh();
    mx=85/MaxDl; my=85/MaxSh;
    mxyIm3=mx;
    if (my<mx)mxyIm3=my;
    CleanIm3();
    for(i=0;i<p; i++)
     {
      GenRamkaIm3(i,XcIm3,YcIm3);
      GraphIm3(KilksPointDet[i],Xd[i],Yd[i],XcDet[i],YcDet[i], XcIm3, YcIm3,
            mxyIm3, i, 2);
      }
   }


//=========================================

 int NumDet(int Y)
  {
   int p;
   p=Y/95;
   return p;
  }
```

```
//=============================================
void __fastcall TForm1::Image3MouseDown(TObject *Sender,
     TMouseButton Button, TShiftState Shift, int X, int Y)
{
float XcE,YcE,XcurF,YcurF,Xmax,Ymax,Xmin,Ymin,XcIm3,YcIm3;
  int j,i;
  CleanIm2();
  NDet=NumDet(Y);
  XcE=Image2->Width/2;
  YcE=Image2->Height/2;

  mx=(ScrollBox2->Width-40)/DlDet[NDet];
  my=(ScrollBox2->Height-40)/ShDet[NDet];
  mxyIm2=mx;
  if (my<mx)mxyIm2=my;

GraphIm2(KilksPointDet[NDet],Xd[NDet],Yd[NDet],XcDet[NDet],YcDet[NDet],Xc
E,YcE,mxyIm2, NDet, 2);
   for (j=0;j<KilksPointDet[NDet]; j++)
  Elipse(Xd[NDet][j],Yd[NDet][j],2,XcDet[NDet],YcDet[NDet],XcE,YcE,mxyIm2);
  ReBuildIm3(XcIm3,YcIm3);
   }

//-----------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
 Fl_Btn3=true;
 Fl_Btn4=false;
 Fl_Btn5=false;
 Fl_Btn6=false;
 Fl_Btn7=false;

}
//=============================================================
==
  int NumPointDet(int n, float X[], float Y[], float XcurF, float YcurF)
   {
   int i,p;
   float Dmin,D;
   p=0;
   Dmin=sqrt(powl(X[0]-XcurF,2)+powl(Y[0]-YcurF,2));
```

```
   for(i=1;i<n;i++)
    {
     D=sqrt(powl(X[i]-XcurF,2)+powl(Y[i]-YcurF,2));
     if(Dmin>D)
      {
       p=i;
       Dmin=D;
      }
     }
     return p;
   }
//-------------------------------------------------
  void DelPoint(int &n, float X[],float Y[],int NP)
  {
   int i;
    if (NP==n-1) NP=0;
     for(i=NP;i<n-1;i++)
     {
      X[i]=X[i+1];
      Y[i]=Y[i+1];
      }
    if (NP==0)
     {
       X[n-2]=X[0];
       Y[n-2]=Y[0];
      }
      n--;
     }

//-------------------------------------------------
 void DelPointDet(float XcurF,float YcurF,float XcE,float YcE)
  {
    int j;
     NPointD=NumPointDet(KilksPointDet[NDet],Xd[NDet],Yd[NDet],XcurF,YcurF);
    //Form1->Button3->Caption=IntToStr(NPointD);
     DelPoint(KilksPointDet[NDet],Xd[NDet],Yd[NDet],NPointD);
     CleanIm2();
     GraphIm2(KilksPointDet[NDet],Xd[NDet],Yd[NDet],XcDet[NDet],YcDet[NDet],
           XcE,YcE,mxyIm2, NDet, 2);
     for (j=0;j<KilksPointDet[NDet]; j++)
      Elipse(Xd[NDet][j],Yd[NDet][j],2,XcDet[NDet],YcDet[NDet],XcE,YcE,mxyIm2);
    }
```

```
//================================================================
  void ChangePoint(int n, float X[],float Y[],int NP, float XNew,float YNew)
  {
     if (NP==0||NP==n-1)
      {
       X[n-1]=XNew;
       Y[n-1]=YNew;
       X[0]=XNew;
       Y[0]=YNew;
      }
     else
      {
       X[NP]=XNew;
       Y[NP]=YNew;
       }
   }
//================================================================
   void ChangePointDetLeft(float XcurF, float YcurF, int X, int Y)
    {

NPointD=NumPointDet(KilksPointDet[NDet],Xd[NDet],Yd[NDet],XcurF,YcurF);
      //Form1->Button4->Caption=IntToStr(NPointD);
      Xq=X;
      Yq=Y;
      Fl_Cor=True;
     }
//В режимі зміни положення вершини фіксує кінцеве значення координати
вершини при натиску праву кнопку миші


//================================================================
   void ChangePointDetRight(float XcurF, float YcurF, float XcE, float YcE)
    {
     int j;
     ChangePoint(KilksPointDet[NDet],Xd[NDet],Yd[NDet],NPointD,XcurF,YcurF);
     CleanIm2();
     GraphIm2(KilksPointDet[NDet],Xd[NDet],Yd[NDet],XcDet[NDet],YcDet[NDet],
           XcE,YcE,mxyIm2, NDet, 2);
     for (j=0;j<KilksPointDet[NDet]; j++)
```

```
Elipse(Xd[NDet][j],Yd[NDet][j],2,XcDet[NDet],YcDet[NDet],XcE,YcE,mxyIm2);
      Fl_Cor=False;
    }

//================================================
//Знаходить проекцію float& Xo,float& Yo точки з коорд инатами float Xc,float
Yc на пряму задану
  //точвками float Xa,float Ya,float Xb,float Yb,
 void ProjectionPointOnLine(float Xa,float Ya,float Xb,float Yb,float Xc,float Yc,
                            float& Xo,float& Yo)
  {
    float A,B,C,k;
     if (Ya==Yb)
      {
        Xo=Xc; Yo=Ya;
      }
     else if (Xa==Xb)
      {
        Xo=Xa; Yo=Yc;
      }
         else
          {
           A=Yb-Ya;
           B=Xa-Xb;
           C=Xb*Ya-Xa*Yb;
           k=-A/B;
           Xo=(C/B+Xc/k+Yc)/(k+1/k);
           Yo=k*Xo-C/B;
          }
    }
//-------------------------------------------------
//Визнаячає чи знаходиться проекція float& Xo,float& Yo точки з коорд инатами
float Xc,float Yc на відрізку прямої заданому
  //точками float Xa,float Ya,float Xb,float Yb
bool PointBelongsLineSegment(float Xa,float Ya,float Xb,float Yb,float Xc,float Yc,
                             float Xo,float Yo,float& D)
  {
    float T1,T2,Q1,Q2;
     if (Xa<Xb)
      {
       T1=Xa; T2=Xb;
```

```
      }
     else
      {
      T1=Xb; T2=Xa;
      }
    if (Ya<Yb)
     {
      Q1=Ya; Q2=Yb;
      }
     else
      {
      Q1=Yb; Q2=Ya;
      }
      if (Xo>=T1&&Xo<=T2&&Yo>=Q1&&Yo<=Q2)
       {
        D=sqrt(pow(Xc-Xo,2)+pow(Yc-Yo,2));
        return True;
        }
       else
        {
        D=0;
        return False;
        }
    }

//-------------------------------------------------------
//Визначає номер сторони апроксимуючого многокутника ( return p;)для деталі,
на якій буде знаходитись додаткова вершина та
// її кординати float& Xt,float& Yt
int NumLineSegment(float XcurF,float YcurF, float& Xt,float& Yt)
  {
    int i,j,p;
    float Xo,Yo,D,Dmin;
    bool Fl;

     for(i=0;i<KilksPointDet[NDet]-1;i++)
     {

ProjectionPointOnLine(Xd[NDet][i],Yd[NDet][i],Xd[NDet][i+1],Yd[NDet][i+1],
                      XcurF,YcurF,Xo,Yo);
```

```
Fl=PointBelongsLineSegment(Xd[NDet][i],Yd[NDet][i],Xd[NDet][i+1],Yd[NDet][i
+1],
                           XcurF,YcurF,Xo,Yo,Dmin);
     if (Fl) break;
    }
    j=i;
    p=i;
    Xt=Xo; Yt=Yo;
   for(i=j+1;i<KilksPointDet[NDet]-1;i++)
    {

ProjectionPointOnLine(Xd[NDet][i],Yd[NDet][i],Xd[NDet][i+1],Yd[NDet][i+1],Xc
urF,YcurF,Xo,Yo);

Fl=PointBelongsLineSegment(Xd[NDet][i],Yd[NDet][i],Xd[NDet][i+1],Yd[NDet][i
+1],
                           XcurF,YcurF,Xo,Yo,D);
     if (Fl&&(D<Dmin))
       {
         Dmin=D;
         p=i;
         Xt=Xo; Yt=Yo;
       }
     }
    return p;
  }
//-------------------------------------------------------------


//-------------------------------------------------
//Додаємо додаткову вершину float Xo, float Yo в масив з координатами int &n,
//float X[],float Y[] на стороні int NVidr
 void AddPoint(int &n, float X[],float Y[],float Xo, float Yo,int NVidr)
  {
   int i;
    for(i=n-1;i>=NVidr+1;i--)
    {
     X[i+1]=X[i];
     Y[i+1]=Y[i];
     }
    X[NVidr+1]=Xo;
```

```
    Y[NVidr+1]=Yo;
    n++;
  }
//-----------------------------------------------------
//Управляюча програма для додавання нової вершини на зовнішній контур деталі
void AddNewPoint(float XcurF,float YcurF,float XcE,float YcE)
 {
  int j,NVidr;
  float Xnew,Ynew;
    NVidr=NumLineSegment( XcurF,YcurF,Xnew,Ynew);
    //Form1->Button5->Caption=IntToStr(NVidr);
    AddPoint(KilksPointDet[NDet],Xd[NDet],Yd[NDet],Xnew,Ynew,NVidr);
    CleanIm2();
    GraphIm2(KilksPointDet[NDet],Xd[NDet],Yd[NDet],XcDet[NDet],YcDet[NDet],
          XcE,YcE,mxyIm2, NDet, 2);
    for (j=0;j<KilksPointDet[NDet]; j++)
     Elipse(Xd[NDet][j],Yd[NDet][j],2,XcDet[NDet],YcDet[NDet],XcE,YcE,mxyIm2);
   }
//-----------------------------------------------------
 void KonturR1(int Nb, int Ne, int& Nr1, float Xr1[],float Yr1[])
 {
  int i;
  for(i=Nb;i<=Ne;i++)
   {
    Xr1[i-Nb]=Xd[NDet][i];
    Yr1[i-Nb]=Yd[NDet][i];
   }
  Nr1=Ne-Nb+1;
 }
//-----------------------------------------------------
void  KonturR2(int Nb, int Ne, int& Nr2, float Xr2[],float Yr2[])
 {
  int i;
  for(i=Ne;i<KilksPointDet[NDet];i++)
   {
    Xr2[i-Ne]=Xd[NDet][i];
    Yr2[i-Ne]=Yd[NDet][i];
   }
  for(i=1;i<=Nb;i++)
   {
    Xr2[KilksPointDet[NDet]-Ne-1+i]=Xd[NDet][i];
```

```
    Yr2[KilksPointDet[NDet]-Ne-1+i]=Yd[NDet][i];
   }
  Nr2=KilksPointDet[NDet]-Ne+Nb;
 }
//=================================================
 void KonturR1_R2(float XcE,float YcE,int& Nr1, float Xr1[],float Yr1[],
                              int& Nr2, float Xr2[],float Yr2[])
  {
    int Nb,Ne;
     if (Fl_Left&&Fl_Right)
       {
         if (NLeft<NRight){Nb=NLeft; Ne=NRight;}
       else {Ne=NLeft; Nb=NRight;}
       Fl_Left=false;
       Fl_Right=false;
       KonturR1(Nb,Ne,Nr1,Xr1,Yr1);
       GraphIm2(Nr1,Xr1,Yr1,XcDet[NDet],YcDet[NDet],
              XcE,YcE,mxyIm2,1,4);
       KonturR2(Nb,Ne,Nr2,Xr2,Yr2);
       GraphIm2(Nr2,Xr2,Yr2,XcDet[NDet],YcDet[NDet],
            XcE,YcE,mxyIm2,2,4);
       Form1->GroupBox2->Visible=true;
       }
  }

//=================================================
void __fastcall TForm1::Image2MouseDown(TObject *Sender,
     TMouseButton Button, TShiftState Shift, int X, int Y)
{
     float XcE,YcE,XcurF,YcurF,Xnew,Ynew,Xt,Yt;
  int j,NV,NVidr;
   int i,p;
   float D,Dmin,Xq,Yq, Dx,Dy;
   bool Fl;
  XcE=Image2->Width/2;
  YcE=Image2->Height/2;
  XcurF=(X-XcE)/mxyIm2+XcDet[NDet];
  YcurF=-(Y-YcE)/mxyIm2+YcDet[NDet];
  if (Fl_Btn3) //вилучення активнорї вершини
   if (KilksPointDet[NDet]>4) DelPointDet(XcurF,YcurF,XcE,YcE);
   else ShowMessage("Кількість вершин повинна бути > 3");
     if (Fl_Btn4)//зміна положення активної вершини
```

```
    {if (Button==mbLeft) ChangePointDetLeft(XcurF,YcurF,X,Y);
    else
     if (Button==mbRight)ChangePointDetRight(XcurF,YcurF,XcE,YcE);}
      if (Fl_Btn5) AddNewPoint( XcurF,YcurF,XcE,YcE);
      if (Fl_Btn6||Fl_Btn7)
     {
       if (Button==mbLeft)
        {NLeft=NumPointDet(KilksPointDet[NDet],Xd[NDet],Yd[NDet],XcurF,YcurF);
         //Form1->Button6->Caption=IntToStr(NLeft);
         Fl_Left=true;}
       if (Button==mbRight)

{NRight=NumPointDet(KilksPointDet[NDet],Xd[NDet],Yd[NDet],XcurF,YcurF);
         //Form1->Button7->Caption=IntToStr(NRight);
       Fl_Right=true;}
       KonturR1_R2(XcE,YcE,Nr1,Xr1,Yr1,Nr2,Xr2,Yr2);
     }


}
//---------------------------------------------------

void __fastcall TForm1::Button4Click(TObject *Sender)
{
 Fl_Btn3=false;
 Fl_Btn4=true;
 Fl_Btn5=false;
 Fl_Btn6=false;
 Fl_Btn7=false;

}
//---------------------------------------------------



void __fastcall TForm1::Button5Click(TObject *Sender)
{
   Fl_Btn3=false;
 Fl_Btn5=true;
 Fl_Btn4=false;
 Fl_Btn6=false;
 Fl_Btn7=false;
```

```
}
//---------------------------------------------------

void __fastcall TForm1::Button6Click(TObject *Sender)
{

 Fl_Btn3=false;
 Fl_Btn6=true;
 Fl_Btn4=false;
 Fl_Btn5=false;
 Fl_Btn7=false;
}
//=====================================================
=================================
void AddTwoKontur(int Nr1,float Xr1[],float Yr1[],int Nr2,float Xr2[],float Yr2[],
int& Nr, float Xr[],float Yr[])
  {
   int i;
    for(i=0;i<Nr1;i++)
     {
      Xr[i]=Xr1[i];
      Yr[i]=Yr1[i];
      }
    for(i=0;i<Nr2;i++)
     {
      Xr[Nr1+i]=Xr2[i];
      Yr[Nr1+i]=Yr2[i];
      }
     Nr=Nr1+Nr2;
  }
//---------------------------------------------------
void BSplayn (int n, int l, float Xr[],float Yr[], int& m, float Xs[],float Ys[])
 {
  int i,j,p;
  float t,h,A0,A1,A2,A3,B0,B1,B2,B3;
  for(i=n-1;i>=0;i--)
   {
    Xr[i+1]=Xr[i];
    Yr[i+1]=Yr[i];
    }
  Xr[0]=Xr[1];
  Yr[0]=Yr[1];
```

```
 n++;
 Xr[n]=Xr[n-1];
 Yr[n]=Yr[n-1];
 n++;
 h=1.0/l;
 p=0;
 Xs[0]=Xr[0];
 Ys[0]=Yr[0];
 for(i=1;i<n-2;i++)
  {
   A0=(Xr[i-1]+4*Xr[i]+Xr[i+1])/6;
   A1=(-Xr[i-1]+Xr[i+1])/2;
   A2=(Xr[i-1]-2*Xr[i]+Xr[i+1])/2;
   A3=(-Xr[i-1]+3*Xr[i]-3*Xr[i+1]+Xr[i+2])/6;
   B0=(Yr[i-1]+4*Yr[i]+Yr[i+1])/6;
   B1=(-Yr[i-1]+Yr[i+1])/2;
   B2=(Yr[i-1]-2*Yr[i]+Yr[i+1])/2;
   B3=(-Yr[i-1]+3*Yr[i]-3*Yr[i+1]+Yr[i+2])/6;
   for(j=0;j<=l-1;j++)
    {
     t=h*j;
     p=p+1;
     Xs[p]=((A3*t+A2)*t+A1)*t+A0;
     Ys[p]=((B3*t+B2)*t+B1)*t+B0;
     }
    }
   p=p+1;
   Xs[p]=Xr[n-2];
   Ys[p]=Yr[n-2];
   m=p+1;
  }
//-----------------------------------------------------
void InterSplayn (int n, int l, float Xr[],float Yr[], int& m, float Xs[],float Ys[])
 {
  int i,j,p;
  float t,h,A0,A1,A2,A3,B0,B1,B2,B3;
  for(i=n-1;i>=0;i--)
   {
    Xr[i+1]=Xr[i];
    Yr[i+1]=Yr[i];
    }
  Xr[0]=Xr[1];
```

```
 Yr[0]=Yr[1];
 n++;
 Xr[n]=Xr[n-1];
 Yr[n]=Yr[n-1];
 n++;
 h=1.0/l;
 p=0;
 Xs[0]=Xr[0];
 Ys[0]=Yr[0];
 for(i=1;i<n-2;i++)
  {
   A0=Xr[i];
   A1=(-Xr[i-1]+Xr[i+1])/2;
   A2=(2*Xr[i-1]-5*Xr[i]+4*Xr[i+1]-Xr[i+2])/2;
   A3=(-Xr[i-1]+3*Xr[i]-3*Xr[i+1]+Xr[i+2])/2;
   B0=Yr[i];
   B1=(-Yr[i-1]+Yr[i+1])/2;
   B2=(2*Yr[i-1]-5*Yr[i]+4*Yr[i+1]-Yr[i+2])/2;
   B3=(-Yr[i-1]+3*Yr[i]-3*Yr[i+1]+Yr[i+2])/2;
   for(j=0;j<=l-1;j++)
    {
     t=h*j;
     p=p+1;
     Xs[p]=((A3*t+A2)*t+A1)*t+A0;
     Ys[p]=((B3*t+B2)*t+B1)*t+B0;
     }
    }
   p=p+1;
   Xs[p]=Xr[n-2];
   Ys[p]=Yr[n-2];
   m=p+1;
 }

 //=============================================
void Splain(float XcE,float YcE)
 {
 int j;
 if (Form1->RadioButton1->Checked)
 {
  if (Fl_Btn6) BSplayn (Nr1,5,Xr1,Yr1,Nr,Xr,Yr);
 if (Fl_Btn7) InterSplayn (Nr1,5,Xr1,Yr1,Nr,Xr,Yr);
  AddTwoKontur(Nr2,Xr2,Yr2,Nr,Xr,Yr,KilksPointDet[NDet],Xd[NDet],Yd[NDet]);
```

```
 }
 else
 {
  if (Fl_Btn6) BSplayn (Nr2,5,Xr2,Yr2,Nr,Xr,Yr);
  if (Fl_Btn7) InterSplayn (Nr2,5,Xr2,Yr2,Nr,Xr,Yr);
  AddTwoKontur(Nr1,Xr1,Yr1,Nr,Xr,Yr,KilksPointDet[NDet],Xd[NDet],Yd[NDet]);
 }
  CleanIm2();
  GraphIm2(KilksPointDet[NDet],Xd[NDet],Yd[NDet],XcDet[NDet],YcDet[NDet],
           XcE,YcE,mxyIm2,2,2);
  for (j=0;j<KilksPointDet[NDet]; j++)
     Elipse(Xd[NDet][j],Yd[NDet][j],2,XcDet[NDet],YcDet[NDet],XcE,YcE,mxyIm2);
}
//=================================================
void __fastcall TForm1::Button8Click(TObject *Sender)
{
  float XcE,YcE;
 int j, Ns;
  if (Fl_Btn6||Fl_Btn7){
  XcE=Image2->Width/2;
  YcE=Image2->Height/2;
  GroupBox2->Visible=false;
  Splain(XcE,YcE);  }
}
//-------------------------------------------------

void __fastcall TForm1::Button7Click(TObject *Sender)
{
 Fl_Btn3=false;
 Fl_Btn7=true;
 Fl_Btn4=false;
 Fl_Btn5=false;
 Fl_Btn6=false;
}
//=================================================
============
 float Delta(float Xa,float Ya,float Xb,float Yb,float Xc,float Yc)
  {
   double A,B,C,D,R1,R2;
     A=Yb-Ya;
     B=Xa-Xb;
     C=Xb*Ya-Xa*Yb;
```

```
    R1=fabs(A*Xc+B*Yc+C);
    R2=powl((A*A+B*B),0.5);
    D=R1/R2;
      return D;
  }
//=================================================

void  SealInformation(int N, float X[], float Y[], float eps,
                int& Nn, float Xn[], float Yn[])
{
 int i,j,p,m,q,t;
 float D;
  i=0; j=1; p=2; m=0; t=0;
  Xn[m]=X[0];
  Yn[m]=Y[0];
while (p<N)
 {
  for(q=j;q<p;q++)
    {
     D=Delta(X[i],Y[i],X[p],Y[p],X[q],Y[q]);
     if (D>eps)
      {
       m++;
       Xn[m]=X[p-1];
       Yn[m]=Y[p-1];
       i=p-1; j=p; p=p+1;t=0;
       break;
       }
    else t++;
   }
   if(t==p-i-1)
    {p++; t=0;}
  }
 Xn[m+1]=X[N-1];
 Yn[m+1]=Y[N-1];
 Nn=m+2;
}

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
void __fastcall TForm1::Button9Click(TObject *Sender)
{
  float XcE,YcE;
```

```
 int j;
  XcE=Image2->Width/2;
  YcE=Image2->Height/2;
  SealInformation(KilksPointDet[NDet],Xd[NDet],Yd[NDet],0.1,
                  KilksPointDet[NDet],Xd[NDet],Yd[NDet]);
  CleanIm2();

GraphIm2(KilksPointDet[NDet],Xd[NDet],Yd[NDet],XcDet[NDet],YcDet[NDet],Xc
E,YcE,mxyIm2,2,2);
  for (j=0;j<KilksPointDet[NDet]; j++)
   Elipse(Xd[NDet][j],Yd[NDet][j],2,XcDet[NDet],YcDet[NDet],XcE,YcE,mxyIm2);
}

//---------------------------------------------------
void __fastcall TForm1::SaveDgt1Click(TObject *Sender)
{
  { char Fname[120],T;
         AnsiString NameF;
         int i,j,L;
if (SaveDialog1->Execute())   {
    NameF=SaveDialog1->FileName;
      for (i=0;i<119;i++)Fname[i]=NULL;
      L=NameF.Length();
       for(i=1; i<=L;i++)
      {    T=NameF[i];
        Fname[i-1]=T; }  }
       ofstream output(Fname,ios::out);
       output<<"Model"<<endl;
       output<<"AAAAA"<<endl;
       output<<KilDet<<endl;
       for(i=0;i<p;i++)
  output<<"Detal"<<i+1<<endl;
 NameDet[p-1]=IntToStr(p);
 for(i=0;i<p;i++)
 output<<KilksPointDet[i]<<endl;
 for(i=0;i<p;i++)
 for(j=0;j<KilksPointDet[i];j++)
 if(j==0)
    output<<Xd[i][j]<<"        "  <<Yd[i][j]<<"   Detal"<<i+1<<endl;
   else output<<Xd[i][j]<<"        "   <<Yd[i][j]<<endl;
   output.close();
```

```
   }
}
//---------------------------------------------------

void __fastcall TForm1::OpenDgt1Click(TObject *Sender)
{ char Fname[120],T,ModelT[40],Q[20],Det[30][20];
 AnsiString NameF;  int i,j,L;
   for (i=1;i<41;i++)ModelT[i]=NULL;
   for (i=1;i<21;i++)Q[i]=NULL;
    for (j=0;j<30;j++)
     for(i=0;i<20;i++) Det[j][i]=NULL;
if (OpenDialog1->Execute())   {
   NameF=OpenDialog1->FileName;
   for (i=0;i<119;i++)Fname[i]=NULL;
   L=NameF.Length();
  for(i=1; i<=L;i++)   {
    T=NameF[i];
    Fname[i-1]=T; }   }
 ifstream input(Fname,ios::in);
 input>>ModelT;
 input>>Q;
input>>KilDet;
for(i=0;i<KilDet;i++)
input>>Det[i];
for(i=0;i<KilDet;i++)
input>>KilksPointDet[i];
for(i=0;i<KilDet;i++)
   for(j=0;j<KilksPointDet[i];j++)
if (j==0)
input>>Xd[i][j]>>Yd[i][j]>>Q;
   else
   input>>Xd[i][j]>>Yd[i][j];
   input.close();
  float XcE,YcE,XcIm3,YcIm3;
   p=KilDet;
   XcE=Image2->Width/2;
   YcE=Image2->Height/2;
   PageControl1->ActivePage=TabSheet2;
   for(i=0;i<p;i++) ParamDet(i);
   ParamModeli();
   BuildIm2(XcE,YcE);
   BuildIm3(XcIm3,YcIm3);
```

```
}  */
//=================================================
void __fastcall TForm1::Dgt1Click(TObject *Sender)
{ char Fname[120],T,ModelT[40],Q[20],Det[30][20];
 AnsiString NameF;  int i,j,L;
   for (i=1;i<41;i++)ModelT[i]=NULL;
   for (i=1;i<21;i++)Q[i]=NULL;
    for (j=0;j<30;j++)
     for(i=0;i<20;i++) Det[j][i]=NULL;
if (OpenDialog1->Execute())   {
   NameF=OpenDialog1->FileName;
   for (i=0;i<119;i++)Fname[i]=NULL;
   L=NameF.Length();
  for(i=1; i<=L;i++)   {
    T=NameF[i];
    Fname[i-1]=T; }   }
 ifstream input(Fname,ios::in);
 input>>ModelT;
 input>>Q;
input>>KilDet;
for(i=0;i<KilDet;i++)
input>>Det[i];
for(i=0;i<KilDet;i++)
input>>KilksPointDet[i];
for(i=0;i<KilDet;i++)
   for(j=0;j<KilksPointDet[i];j++)
if (j==0)
input>>Xd[i][j]>>Yd[i][j]>>Q;
   else
   input>>Xd[i][j]>>Yd[i][j];
   input.close();
  float XcE,YcE,XcIm3,YcIm3;
   p=KilDet;
   XcE=Image2->Width/2;
   YcE=Image2->Height/2;
   PageControl1->ActivePage=TabSheet2;
   for(i=0;i<p;i++) ParamDet(i);
   ParamModeli();
   BuildIm2(XcE,YcE);
   BuildIm3(XcIm3,YcIm3);
}
//---------------------------------------------------
```