МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розробка математичного та програмного забезпечення для побудови щільних укладок плоских об'єктів, близьких до опуклих

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Виконав: студент групи МгІТ-2-22

Тарас МАМОНОВ

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

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

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерні науки

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

_____Володимир ЩЕРБАНЬ

«___»_____2023___року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА

Мамонову Тарасу Андрійовичу

1. Тема роботи: Розробка математичного та програмного забезпечення для побудови щільних укладок плоских об'єктів, близьких до опуклих. Науковий керівник роботи д.т.н., професор Чупринка Віктор Іванович, затверджені наказом закладу вищої освіти від 12 . 09.2023 року , № 210-уч.

2. Вихідні дані до кваліфікаційної роботи: Розробка кафедри комп'ютерних наук

3. Зміст кваліфікаційної роботи (перелік питань, які потрібнорозробити)

Вступ, розділ 1(САПР взуття); розділ 2 (Математичне забезпечення для побудови щільних укладок плоских об'єктів, близьких до опуклих); розділ 3(Програмне забезпечення для побудови щільних укладок плоских об'єктів, близьких до опуклих). Додатки – програмні коди модулів системи.

4. Дата видачі завдання 08.2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної магістерської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	30.08.2023	
2	Розділ 1 САПР взуття	06.09.2023	
3	Розділ 2. Математичне забезпечення для побудови щільних укладок плоских об'єктів, близьких до опуклих	28.09.2023	
4	Розділ 3. Програмне забезпечення для побудови щільних укладок плоских об'єктів, близьких до опуклих	21.10.2023	
5	Висновки	29.10.2023	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	06.11.2023	
7	Подача кваліфікаційної роботи (проєкту) науковому керівнику для відгуку (за 14 днів дозахисту)		
8	Подача кваліфікаційної роботи (проєкту) для рецензування		
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату		
10	Подання кваліфікаційної роботи на затвердження завідувачу кафедри		

З завданням ознайомлений:

Студент _____ Тарас МАМОНОВ
(підпис) (ім'я та прізвище)

Науковий керівник роботи _____ Віктор ЧУПРИНКА
(підпис) (ім'я та прізвище)

АНОТАЦІЯ

Мамонов Т.А. Розробка математичного та програмного забезпечення для побудови щільних укладок плоских об’єктів, близьких до опуклих – Рукопис.

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 побудови щільних укладок плоских об’єктів, близьких до опуклих. Запропоновані алгоритми реалізовані в програмний продукт для автоматизованої побудови щільних укладок плоских об’єктів, близьких до опуклих.

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, щільні укладки, плоскі об’єкти

ABSTRACT

Mamonov T.A. Development of mathematical and software for constructing dense stacks of flat objects close to convex ones - Manuscript.

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

The paper proposes methods and algorithms for constructing dense stacks of flat objects close to convex ones. The proposed algorithms are implemented in a software product for the automated construction of dense stacks of flat objects close to convex ones.

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

Keywords: mathematical and software, dense stacking, flat objects

Зміст

ВСТУП

Ринок товарів на замовлення постійно зростає, все більше у всьому світі стає людей зацікавлених у придбанні продукції на замовлення, яка б відповідала їхнім індивідуальним потребам, смакам, соціальному статусу чи рангу. Взуття служить для полегшення пересування та запобігання травмам стопи.

Проте, вибір доступного взуття має деякі потенційно суперечливі критерії для споживачів, де доводиться часто вибирати між комфортом і естетикою. Більшість, як правило, віддають перевагу візуально-естетичним показникам над комфортним взуттям, але нажаль цей вибір в результаті призводить до різноманітних паталогій стопи в процесі носіння виробу та після. Крім того, більшість виробників взуття зараз імітують сучасні модні стилі у виробах, але йдуть на компроміс з якістю, щоб знизити вартість товару.

Взуття повинно відповідати низці вимог, виконувати захисну, утилітарну та естетично-ергономічні функції, оскільки це дозволить створити оптимальні умови для нормальної діяльності стопи та організму в цілому. Естетично-ергономічні функції дозволяють створити природне середовище для стопи без її деформацій, тобто, за формою та окремими абрисами взуття має відповідати сформованій стопі. Певною мірою забезпечення естетично-ергономічних функцій взуття залежить і від якості проектування, тому що рационально спроектоване взуття з врахуванням анатомо-морфологічних властивостей стопи забазпечує раціональність даного виробу у подальшій його експлуатації.

Протягом тисячоліть ремісництво взуття завжди має свій попит та відіграє важливу роль у суспільстві. Виробники взуттєвих підприємств, студії та ательє пропонують пошив індивідуального взуття для клієнта. Майстри взуття задовольняють конкретні запити персоналізованої продукції споживача.

При класичному методі проектування взуття на колодку наносять допоміжні і стильові лінії, а потім за цими контурами проектують деталі. Таким чином ми визначаємо частину тривимірної поверхні колодки, потім

«розплющуємо» її для отримання плоских шаблонів і викроювання деталей, а потім заготовку, зібрану з плоских деталей, затягуємо на колодку, перетворюючи їй знову в просторовий об’єкт. Така подвійна трансформація часто призводить до того, що отриманий виріб не зовсім відповідає задуму модельєра.

Постійно зростаючі вимоги до дизайну взуття, його якості при одночасній необхідності скорочення термінів розробки нових моделей і їхнього запуску в серійне виробництво, а також утримання цін на конкурентоздатному рівні в буквальному значенні, змушують виробників впроваджувати новітні технології на всіх етапах проектування та виготовлення взуття. Безпосереднє перенесення зарубіжного досвіду, технологій, устаткування не завжди прийнятне, оскільки, вимагає значних капіталовкладень, практично недоступних для сучасного вітчизняного підприємства. Розробка власних систем автоматизованого проектування в першу чергу вимагає попередньої розробки їх наукової аналітичної основи, спираючись при цьому на концепцію виробництва, характерного для масового, але враховуючи при цьому анатомо-морфологічні властивості стоп.

В сучасних умовах швидкої зміни моди і смаків споживача та наявності великого асортименту взуття, прискорюється процес морального старіння виробів, тому важливо, щоб темпи їх проектування були якнайшвидшими. Для конкурентоспроможності виробів взуттєвого виробництва і підвищення його ефективності першочергове значення має рівень конструкторсько-технологічної підготовки, якої сприяє застосування систем автоматизованого проектування (САПР). За останній час у нас в країні, як і в усьому світі, здійснюється комп'ютеризація взуттєвого виробництва, активно впроваджуються нові інформаційні технології.

Методи дослідження. Теоретичні дослідження роботи грунтуються на комплексі основних положень виробницва взуття та застосування 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 проєктування декоративних елементів на деталях взуття.

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 проєктування декоративних елементів на деталях взуття» опубліковано одна наукових робота.

Структура та обсяг магістерської роботи. Магістерська робота містить вступ, три розділи, загальні висновки досліджень, список використаних джерел, додатки.

1. САПР ВЗУТТЯ

1. 1. Оглядова характеристика САПР взуття

Система автоматизованого проектування (САПР) взуття являє собою організаційно-технічну систему, що складається з комплексу коштів автоматизованого проектування взаємодіючого з розробниками проектно-конструкторської документації.

САПР реалізовується в різних варіантах проєктування[1-3]:

1. Просторове проектування. У його основу встановлені алгоритми, що забезпечують роботу пристроїв знімання параметрів поверхні взуттєвої колодки і отримання умовної розгортки для розробки плоских деталей конструкції (система 3D). Алгоритми просторового проектування на вітчизняних і зарубіжних підприємствах не знаходять широкого практичного застосування через складність і високу вартість таких пристроїв.

2. Площинне проектування. Ведучою є концепція розробки і вдосконалення алгоритмів функціонування автоматизованих систем площинного проектування (система 2D). Структура побудови такого програмного комплексу базується на двох основних принципах:

- спадкоємність і органічна єдність з діючими структурами підготовки виробництва;
- об'єднання і рішення максимально можливого числа задач конструкторської підготовки з урахуванням можливостей комп'ютерної техніки.

Аналіз чого склався структури процесу підготовки проекту і потреби виробництва передбачає наявність в системі проектування наступних базових програмних модулів:

- введення початкової інформації;
- проектування моделі і її деталей;
- градирование контурів деталей;
- виведення на графічний або друкуючий пристрій;

- контроль укладиваемости деталей;
- розрахунок трудомісткості зборки моделі;
- формування паспорта моделі.

Пакет прикладних програм являє собою комплекс програм, працюючий під управлінням головної програми і призначений для рішення певного класу задач, як правило, близьких за змістом або по вживаних математичних методах. Пакет прикладних програм є найбільш довершеною формою програмного забезпечення САПР.

Системи і програмні комплекси автоматизованого проектування взуття: Уперше САПР взуття успішно демонструвалися в Пірмазенсе (ФРН), а потім на виставці "Тиждень шкіри" в Парижі в 1985 році. Розробки САПР знайшли підтримку у ведучих взуттєвих фірм, що надали свої підприємства для випрбовування експериментальних систем.

Двухкоординатние системи автоматизованого проектування на всіх підприємствах зарекомендували себе добре, хоч і з різним рівнем продуктивності. Проте, сам вибір системи для конкретного підприємства залишається досить складним, оскільки необхідний, щоб системи відповідали технічним умовам виробництва

Трехкоордінатние системи автоматизованого проектування (об'ємного) ще знаходяться на стадії становлення. При створенні моделі взуття основний акцент робиться на отриманні малюнка, що відображає задум художника-модельєра. Конструювання ж вимагає геометрично точного зображення реальної моделі. Отримують зображальну інформацію і геометричні дані в трьохмірному уявленні. Якщо САПР здатна представляти зображальну інформацію, то систему називають ескізною, зображальною, графічною. Якщо САПР дозволяє отримувати геометричні дані, то її можна використати для управління роботою обладнання.

Так, отримані з допомогою САПР геометричні дані, необхідні для розрахунку траєкторії руху робочих органів взуттєвих машин, використовуються в системі автоматизованого виробництва. За рубежем використовується багато систем CAD, серед них: Gradamatic, Apex - фірми Camsco (США). У останніх варіантах цих систем виконують наступні операції:

- конструювання деталей;
- градирование деталей;
- оцінка моделей, аналіз економічності (вартість і трудомісткість);
- визначення площ;
- калькуляція витрати матеріалів і заробітної плати.

До числа найбільш відомих і поширених відносяться:

Система FDS Microdynamics (США)[4]

Система проектування взуття FDS розроблена як серія модулів. Завдяки своїй конструкції FDS може охоплювати 1000 робочих місць або АРМ. У FDS кожне робоче місце має свій комп'ютер і запам'ятовуючий пристрій. Все це тісно пов'язане з іншими робочими місцями і центром. Існує модель системи FDS -50 для двухразмерного моделювання нових фасонів взуття швидких зарисовок.

Модель FDS -1500 дозволяє проектувати, конструювати, градировать і робити укладиваемость деталей в двомірних координатах. Модель FDS -300 і FDS -350 - трьохмірні проектуючі системи.

Система фірми Clarks

Система базується на автоматичному описі складної поверхні колодки з використанням графічних пристроїв з числовим управлінням. Роботи, проведені на фірмі, показали, що прийнята сьогодні система двухразмерного градирования приводить до значних витрат матеріалу. Тому фахівці фірми розробили систему, яка градирует заготівлі в трьох напрямах. Цей метод використовується у Франції. Фірма USM (Англія) створила систему CAD-CAM, яка дозволяє по числовій

інформації за допомогою термінала отримувати креслення деталей. Фірма розробила три методи підготовки зразка.

У одному модельєр вичерчивает модель зразка в двухкоординатной сітці на моніторі ЕОМ, що включає телевізійний екран, креслярський програматор, що має в своїй пам'яті 16,7 млн різних колірних відтінків, ліній, дуг.

У другому методі дані про взуття вводяться в ЕОМ за допомогою телевізійної камери. Оператор може задати колір, комбінацію матеріалів і т. д. У третьому методі обробляється інформація в трьохмірному вимірюванні. Трьохмірний координатор знімає координати з колодки, ЕОМ тут же дає точну копію. На цьому трьохмірному зображенні колодки конструктор на екрані дисплея зображає деталі взуття. Далі конструктор використовує інформацію про матеріали. Комп'ютер розраховує площі деталей, визначає економічність і т. д.

Система фірми Lectra[5]

З середини 80-х років більше за 300 систем фірми Lectra працюють у взуттєвій промисловості Франції, Англії, ФРН, Італії, США. Таким же чином фірма Dic & Jordan спільно з Bata Engineering створила систему підготовки трьохмірних зразків, які використовуються для контролю виробів при виготовленні різаків і в інших операціях САМ. Система BATA конструює колодку і взуття також в трьохмірному вимірюванні. З останніх розробок потрібно виділити Digiton (Канада), Crispin (Австрія), Cixi (Франція).

Системи виконують наступні функції:

- введення даних про колодку;
- моделювання взуття в інтерактивному режимі;
- отримання розгортки і деталировка верху;
- градирование деталей;
- розміщення деталей взуття на шкірі;
- виготовлення шаблонів деталей верху взуття.

Процес закінчується виготовленням лекал деталей взуття за допомогою лазерного автомата.

Система Apex

Розроблена фірмою Camsco (США).

Система передбачає наступні операції:

- знімання інформації про перетин колодки;
- запис в цифровій формі відповідних ліній моделі, нанесених на колодці;
- розрахунок і отримання на екрані графічного дисплея умовних розгорток зовнішніх і внутрішніх бічних поверхонь колодки;
- розмноження деталей (градирование);
- виготовлення креслень і підготовка документації;
- вирізування шаблонів для моделювання.

Крім того, система може виконувати перехідні операції від антропометричних даних до розмірів деталей взуття. Вона забезпечує проектування взуття на основі способу жорсткої оболонки. Система по певному коду викликає з пам'яті машини УРК потрібного фасону, а також типові деталі верху. Потім на екрані графічного дисплея будується система координат, відкладаються по завданню конструктора-програміста необхідні кути розведення крил, в які автоматично вписується УРК, будуються контури деталей верху і відкладаються величини затяжної кромки і інші параметри. Система забезпечує проектування окремих деталей верху з припуском на обробку, а також креслення внутрішніх і проміжних деталей, серійну градирование деталей.

1.2. Структура і характеристика модульної САПР "Ірис"

Програмний комплекс розроблений на кафедрі КТИК КНУТД в 1990-2005 рр. і за цей час впроваджений на багатьох взуттєвих підприємствах України різної форми власності[6-8].

По своєму характеру система є системою площинного проектування 2D.

Програмний комплекс складається з декількох програмних модулів, які об'єднуються під егідою головного меню:

- оцифровка контурів;
- проектування;
- градирование подетальное;
- градирование грунтів;
- вичерчивание картинок;
- розміщення шаблонів;
- каталог;
- інструкція користувача.

Структура комплексу

Структура побудови комплексу такого роду базується на двох основних принципах:

- спадкоємність і єдність з діючими структурами конструкторської підготовки взуттєвого виробництва;
- об'єднання як можна більшого числа задач в конструкторській підготовці виробництва.

Програмний комплекс призначений для проетирования деталей верху і низу взуття різних конструкцій. Аналіз чого склався структури процесу виробництва і підготовки проекту передбачає наявність в системі наступних базових програмних модулів:

- введення початкової інформації;
- проектування моделі і її деталей;
- серійне градирование контура деталей;
- контроль укладиваемости контура деталей;
- формування паспорта моделі.

Функції автоматизованого проектування на програмному комплексі "ІРИС". Широке застосування методів автоматизованого проектування взуття і кожгалантерейних виробів передбачає використання послідовності цілеспрямованих функцій. Під функцією автоматизованого проектування в цьому випадку розуміємо специфічний вплив системи на об'єкт проектування, направлений на приведення його до вигляду, відповідного вимогам конкретного етапу розробки. У результаті реалізації необхідних функцій отримують комплект конструкторської документації, необхідний для запуску виробу у виробництво.

Основу автоматизованого проектування складають прийоми звичайних (креслярських) методів, що отримали свій природний розвиток.

Жодна з сучасних систем не може вважатися завершеною, якщо вона не в змозі виконати хоч би одна вимога повсякденної практики розробки конструкцій взуття. Якісно новий рівень процесу проектування повинен включати досить широкий спектр додаткових можливостей, що надаються модельєру-конструктору. Тому цілком закономірним є розділення функцій автоматизованого проектування на традиційні і специфічні (машинні).

Традиційні функції формуються з набору функцій, необхідних для підготовки графічних і текстових документів. Графічні поділяються на функції прямої і послідовної дії. Перші характеризуються досягненням результату безпосередньо після їх ініціалізації. Функції послідовної дії вимагають введення додаткових даних, вказуючих на інтенсивність і область поширення. У залежності від масштабу додатку функції прямої дії поділяються на параметричні, одиничні і структурних.

Використання параметричних функцій дозволяє визначати габарити об'єкта{G}(деталі, декількох деталей або вузлів, грунд-моделі), довжину окремих контурів{x}, периметр {AX}, площа {z}, відстань {X} між заданими точками, кут нахилу вибраних ліній {y}. Результати висвічуються на екрані або зберігаються в файлі. Одиничні функції прямої дії дають можливість перемістити будь-яку

точку в нове положення {}, встановити першу точку {T}, видалити {U}, сдублировать {w} або вставити {W} нову точку на заданій відстані. Одночасно можна задати потрібний напрям {h} точок по замкненому контуру.

Структурні функції виконують дії над вказаними об'єктами (замкненими або розімкненими контурами, окремими лініями, вставками і т. п.). Зсув {F6} - переміщення об'єкта у вказаному напрямі відносно всього зображення.

Функції прямої дії ініціалізувалися однократним натисненням відповідної клавіші або переміщенням курсора в задану область екранного меню.

Функції послідовної дії передбачають багаторазове натиснення клавіш з вказівкою необхідних параметрів і поділяються на елементарні, об'єктні і функції формообразования. Елементарні функції впливають на окремі дільниці (елементи) контура. Згладжування {L} дозволяє вирівнювати значення координатних пар методом найменших квадратів у вказаному інтервалі.

Функція інтерполяції {I} дозволяє визначити бракуючі точки плавного контура за допомогою інтерполяційних многочленів (наприклад, сплайнов), зберігаючи при цьому координати вузлових (початкових) точок.

Функції припускІв {e} і {v} призначені для проведення постійних (під строчку, загибку і т. п.) і змінних (затяжна кромка) припусков.

Об'єктні функції виконують перетворення з готовими фрагментами деталей. Функція з'єднання {s} дозволяє з'єднати два різних контури в одну деталь з привласненням імені.

Площа при цьому визначається по зовнішньому контуру.

Функція вставки {F} є однією з найбільш могутніх в програмному комплексі і дозволяє сформувати деталь з окремих дільниць декількох контурів.

Корисно застосовувати цю функцію при введенні початкової інформації на дигитайзере, оскільки вельми скрутно ввести в комп'ютер абсолютно однаково співпадаючі контури двох і більш деталей. Для підготовки складального креслення моделі взуття із зовнішніми і внутрішніми берцами і задинками

включають функцію {r}, що дозволяє розвертати деталь навколо заданої осі (лінії перегину).

Особливу групу складають функції формообразования, що дозволяють створювати окремі елементи моделі, наприклад, за допомогою малювання замкнених або розімкнених контурів {q}. Мітки для зборки або накол {i}, мітки для бло-чек {до}, для перфорації можуть виконуватися як окремі деталі {I},{До},{Р} з привласненням імені. За допомогою функції геометричних фігур {F7} можна створювати кола, овали, багатокутники і лінії з різними геометричними параметрами (радіус, довжина, ширина) і під різними кутами.

Машинні функції дозволяють створити особливу оболонку і надають певний сервіс модельєру, який може підібрати зручний режим роботи.

Функціями управління екраном масштабують зображення {F10}, зсувають його в зручну позицію, очищають екран {ESC}, викликають поточну підказку і меню {F1}. Сюди ж відноситься функція {M} вибору певної кількості деталей, що одночасно висвічуються на екрані (від однієї до всіх). Вона також міняє порядок виклику окремих деталей.

Функції управління друком дозволяють підібрати шрифт для виведення текстових, визначати розмір тексту і форматувати його по правому або лівому полю, будувати рамки таблиць. Функції управління графопостроителем (плоттер) дають можливість підбору кольору (пера) або типу лінії, вибирати масштаб і месторасположение тексту.

Практичний досвід експлуатації програмного комплексу "ІРИС" показує, що найбільш ефективні результати досягаються при роботі в певній послідовності:

коректування початкових контурів моделі. Контури повинні бути представлений:

- плавними замкненими кривими. Точки розташовуються за годинниковою стрілкою, частота їх залежить від кривизни дільниці. Здвоєні, строєні і т. д. точки не рекомендуються, їх кількість повинна бути мінімальною;

- побудова відрізних деталей. Початкові контури розрізаються на частині по зазделегідь відмічених точках;

- побудова суцільних деталей. Зовнішню сторону деталей грунд-моделі рекомендується розташовувати знизу, внутрішню - зверху;

- побудова технологічних припусков міток. Після установки припусков за допомогою вказаних функцій бажане невелике коректування в кінцевих точках;

- побудова внутрішніх і проміжних деталей. Їх будують, виходячи з контурів зовнішніх деталей. Після побудови віддаляються зайві або здвоєні точки;

- остаточна доробка контура. Перевіряються всі розроблені контури. Проставляються мітки для блочек, перфорації і т. п. Грунд-модель розвертається у вісь градирования;

- виведення на плоттер і збереження файлів. Розроблена модель записується в файл і може бути виведена на плоттер або принтер.

Висновки до першого розділу

Економічність моделей взуття визначається щільністю укладок деталей вузття. Тому розробка математичного та програмного забезпечення для побудови щільних укладок плоских об’єктів, близьких до опуклих, є актуальною задачею при проєктуванні нових моделей взуття.

2. РОЗРОБКА МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРОЄКТУВАННЯ ДЕКОРАТИВНИХ ЕЛЕМЕНТІВ НА ДЕТАЛЯХ ВЗУТТЯ

2.1. Решітчасті укладки

Розглянемо на площині об'єкти S_1 та S_2. Нехай $\text{int } S = S - S^{\wedge}$, де $S^{\wedge}$ – границя об'єкта S. Об'єкти S_1 та S_2 не перетинаються, якщо

$$\text{int } S_1 \cap \text{int} S_2 = 0 \qquad (2.1).$$

Якщо одночасно виконується умова

$$S_1 \cap S_2 \neq 0 \qquad (2.2),$$

то об'єкти S_1 та S_2 називаються щільно розміщеними.

Щільно розміщені об'єкти не мають спільних внутрішніх точок, але обов'язково мають спільні граничні точки.

Система об'єктів S_i, $i=1..p$, утворюють на площині укладку, якщо для кожної пари об'єктів із цієї системи виконуються умови їх взаємного неперетину:

$$\text{int } S_n \cap \text{int } S_m = 0,\ n \neq m,\ n,m=1..p \qquad (2.3)$$

та для будь-якого об'єкта S_i, $i=1..p$ знайдеться хоч один об'єкт S_q, де $q \in [1..p]$, $p \neq i$, який дотикається до об'єкта S_i.

Позначимо через $S+\boldsymbol{a}$ об'єкт, який можна отримати переміщенням кожної точки об'єкта S на вектор $\boldsymbol{a}$ та назвемо його трансляцією об'єкта S.

Множину векторів виду[9-11]

$$\boldsymbol{r} = n\boldsymbol{a_1} + m\boldsymbol{a_2},\ \text{де } n,m=0, \pm1, \pm2, \pm3, \ldots \pm k\ldots \qquad (2.4),$$

де $\boldsymbol{a_1}(a_{1x}, a_{1y})$, $\boldsymbol{a_2}(a_{2x}, a_{2y})$ - лінійно-незалежні вектори, назвемо решіткою з базисом $\boldsymbol{a_1}$, $\boldsymbol{a_2}$ та позначимо через $\Lambda = \Lambda(\boldsymbol{a_1}, \boldsymbol{a_2})$.

Абсолютна величина визначника, який складений із векторів решітки, називається визначником решітки Λ та позначається $\det\Lambda$, де [261]:

$$\det\Lambda = |[\boldsymbol{a_1} \times \boldsymbol{a_2}]| = \begin{vmatrix} a_{1x} & a_{1y} \\ a_{2x} & a_{2y} \end{vmatrix} = \begin{vmatrix} a_{1x} a_{2y} - a_{2x} a_{1y} \end{vmatrix}. \qquad (2.5)$$

Розглянемо систему об'єктів $\bigcup_{n.m} S^{nm}$, де $n,m=0, \pm1, \pm2, \pm3,\ldots\pm k\ldots$, які складаються із трансляції $S^{nm}=S+n\boldsymbol{a_1}+m\boldsymbol{a_2}$ об'єкта S на вектори решітки $\Lambda=\Lambda(\boldsymbol{a_1},\boldsymbol{a_2})$. Якщо ця система є укладкою, то така укладка називається укладкою об'єкта S, виконаної по решітці $\Lambda=\Lambda(\boldsymbol{a_1},\boldsymbol{a_2})$. Решітка $\Lambda=\Lambda(\boldsymbol{a_1},\boldsymbol{a_2})$ в цьому випадку є допустимою для укладки об'єкта S.

Щільність $\delta_s(\Lambda)$ решітчастої укладки можна характеризувати за допомогою співвідношення:

$$\delta_s(\Lambda)=|S| /\det\Lambda , \qquad (2.6),$$

де $|S|$ - площа плоского геометричного об'єкта S, $\det\Lambda$ - визначник решітки $\Lambda=\Lambda(\boldsymbol{a_1},\boldsymbol{a_2})$, за якою виконана укладка. Із наведеного співвідношення видно, що щільність $\delta_s(\Lambda)$ решітчастої укладки тим вища чим менша площа паралелограма, сторонами якого є базові вектори решітки $\boldsymbol{a_1}$ та $\boldsymbol{a_2}$.

Множину векторів виду:

$$\boldsymbol{r_1}=n\boldsymbol{a_1}+m\boldsymbol{a_2} \text{ та } \boldsymbol{r_2}=n\boldsymbol{a_1}+m\boldsymbol{a_2}+\boldsymbol{g}, \text{ де } n,m=0, \pm1, \pm2, \pm3,\ldots\pm k\ldots ,$$

де $\boldsymbol{a_1}$, $\boldsymbol{a_2}$- лінійно-незалежні вектори, назвемо подвійною решіткою з базисом $\boldsymbol{a_1}$, $\boldsymbol{a_2}$ і вектором зсуву решітки $\boldsymbol{g}$ та позначимо через $\boldsymbol{W}=\boldsymbol{W}(\boldsymbol{a_1},\boldsymbol{a_2},\boldsymbol{g})$. Абсолютна величина визначника, який складений із базових векторів подвійної решітки, називається визначником решітки та позначається $\det \boldsymbol{W}$.

Розглянемо систему об'єктів $\bigcup_{n.m} S_1^{nm}$ та $\bigcup_{n.m} S_2^{nm}$, де $n,m=0, \pm1, \pm2, \pm3,\ldots\pm k\ldots$, які складаються із трансляції $S_1^{nm}=S_1+n\boldsymbol{a_1}+m\boldsymbol{a_2}$ та $S_2^{nm}=S_2+n\boldsymbol{a_1}+m\boldsymbol{a_2}+\boldsymbol{g}$ об'єктів S_1 та S_2 на вектори подвійної решітки $\boldsymbol{W}=\boldsymbol{W}(\boldsymbol{a_1},\boldsymbol{a_2},\boldsymbol{g})$ [118]. Якщо ця система є укладкою, то така укладка називається укладкою об'єктів S_1 та S_2, виконаної по подвійній решітці $\boldsymbol{W}=\boldsymbol{W}(\boldsymbol{a_1},\boldsymbol{a_2},\boldsymbol{g})$. Подвійна решітка $\boldsymbol{W}=\boldsymbol{W}(\boldsymbol{a_1},\boldsymbol{a_2},\boldsymbol{g})$ в цьому випадку є допустимою для укладки об'єктів S_1 та S_2. Фрагмент подвійної решітчастої укладки представлений на рис. 3.2. В цьому випадку під об'єктом S_1 мають на увазі деталь S(0) у вихідному положенні, а під об'єктом S_2 мають на

увазі деталь S(π) , повернуту на 180^0 відносно вихідного положення. Подвійна решітка представляє собою дві однакові одинарні решітки $\Lambda_1=\Lambda(\boldsymbol{a_1},\boldsymbol{a_2})$ та $\Lambda_2=\Lambda(\boldsymbol{a_1}+\boldsymbol{g}\ ,\ \boldsymbol{a_2}+\boldsymbol{g})$, які зміщені одна відносно іншої на вектор зсуву решітки $\boldsymbol{g}$. У вузлах решітки Λ_1 розміщаються об'єкти S_1, а у вузлах решітки Λ_2 розміщаються об'єкти S_2.

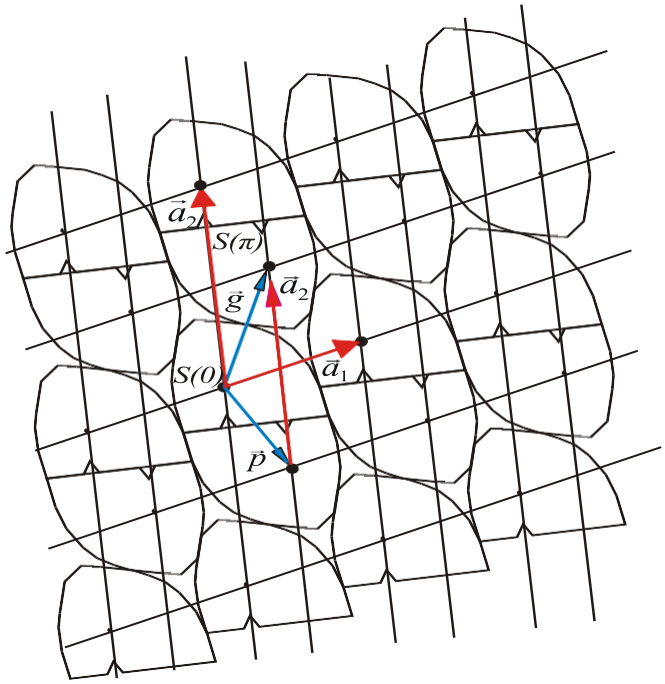

Рис. 2.1. Фрагмент подвійної решітчастої укладки

Абсолютна величина визначника, який складений із векторів решітки, називається визначником решітки $\boldsymbol{W}$ та позначається det $\boldsymbol{W}$, де:

$$\det \boldsymbol{W} = |[\ \boldsymbol{a_1} \times \boldsymbol{a_2}\ ]| = \begin{vmatrix} a_{1x} & a_{1y} \\ a_{2x} & a_{2y} \end{vmatrix} = \begin{vmatrix} a_{1x} & a_{2y} \end{vmatrix} - \begin{vmatrix} a_{2x} & a_{1y} \end{vmatrix}. \qquad (2.7)$$

Щільність $\delta_s(\boldsymbol{W})$ решітчастої укладки можна характеризувати за допомогою співвідношення:

$$\delta_s(\boldsymbol{W}) = (|S_1| + |S_2|) / \det \boldsymbol{W}\ , \qquad (2.8)$$

де $|S_1|$ та $|S_2|$ - відповідно площі плоского геометричного об'єкта S_1 та S_2, det $\boldsymbol{W}$ - визначник решітки $\boldsymbol{W}=\boldsymbol{W}(\boldsymbol{a_1}, \boldsymbol{a_2}, \boldsymbol{g})$, за якою виконана укладка. Із наведеного

співвідношення видно, що щільність $\delta_s(\boldsymbol{W})$ решітчастої укладки тим вища, чим менша площа паралелограма, сторонами якого є базові вектори решітки $\boldsymbol{a_1}$ та $\boldsymbol{a_2}$.

На основі технологічної постановки задач проектування щільних укладок в паралелограмі сформулюємо математичні постановки цих задач.

Математична постановка задачі „Укладка А". Серед множини допустимих решіток $\Lambda^i=\Lambda(\boldsymbol{a}^i_1, \boldsymbol{a}^i_2)$, де $i=1,2..q,$ для побудови щільних укладок для однакових та однаково орієнтованих плоских геометричних об'єктів S знайти таку решітку $\Lambda^*=\Lambda(\boldsymbol{a}^*_1, \boldsymbol{a}^*_2)$, для якої $\delta_s(\Lambda^*)=|S| /\det\Lambda^*=max(\delta_s(\Lambda^i))$, де |S| - площа плоского геометричного об'єкту S . Так як площа плоского геометричного об'єкту S є постійною величиною, то серед допустимих решіток $\Lambda^i=\Lambda(\boldsymbol{a}^i_1,\boldsymbol{a}^i_2)$ необхідно знайти таку, для якої $\det\Lambda^*=min(\det\Lambda^i)$[12-13].

Математична постановка задачі „Укладка Б". Серед множини допустимих подвійних решіток $\boldsymbol{W}^i=\boldsymbol{W}(\boldsymbol{a}^i_1,\boldsymbol{a}^i_2,\boldsymbol{g}^i)$, де $i=1,2..q,$ для побудови щільних укладок для однакових плоских геометричних об'єктів S з поворотом в рядах на 0^0 та 180^0, знайти таку решітку $\boldsymbol{W}^*=\boldsymbol{W}(\boldsymbol{a}^*_1, \boldsymbol{a}^*_2,\boldsymbol{g}^*)$, для якої $\delta_s(\boldsymbol{W}^*)=|S| /\det\boldsymbol{W}^*=max(\delta_s(\boldsymbol{W}^i))$, або $\det\boldsymbol{W}^*=min(\det\boldsymbol{W}^i)$[14-15].

2.2. Аналітичний опис зовнішнього контуру деталі та її положення на площині

Для однозначного відображення положення деталі S в укладці та генерування множини допустимих щільних укладок необхідно аналітично описати зовнішній контур деталі та визначити параметри, які б однозначно відображали положення деталі на площині[14-16]. Вже відмічалось в попередніх розділах, що контури деталей взуття мають складну форму зовнішнього контуру та описати їх аналітично у вигляді математичної функції $F(x,y)=0$ у більшості випадків неможливо. Тому ми будемо апроксимувати деталі S у вигляді многокутників S_m із заданої точністю ε. Для однозначного визначення

зовнішнього контуру многокутника S_m достатньо знати координати вершин $A_i(Xm_i,Ym_i)$, де $i=1,2...n$ та $Xm_1= Xm_n, Ym_1=Ym_n$(рис. 2.2).

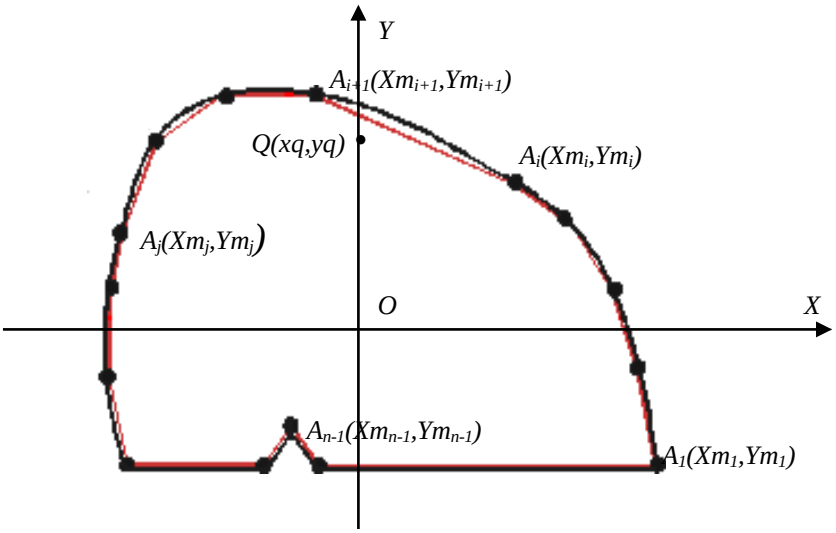

Рис. 2.2. Апроксимація зовнішнього контуру деталі

Тоді координати будь-якої точки $Q(xq,yq)$ на стороні A_iA_{i+1} зовнішнього контуру апроксимуючого многокутника можна представити наступним чином:

$$\begin{cases} xq = (Xm_{i+1} - Xm_i)t_i + Xm_i \\ yq = (Ym_{i+1} - Ym_i)t_i + Ym_i \end{cases}, \quad \text{де } i=1,2...n-1 \text{ та } t_i \in [0,1]. \qquad (2.9)$$

Тобто за допомогою виразу (2.9) можна однозначно аналітично описати зовнішній контур деталі із заданою точністю ε.

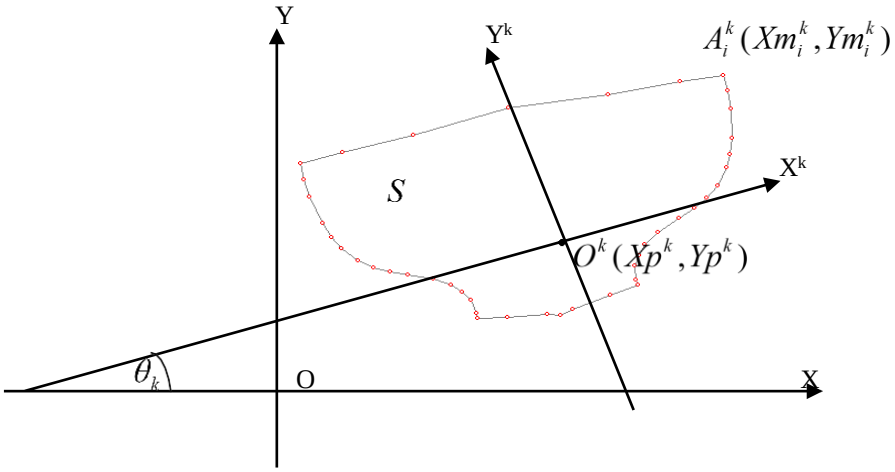

Рис. 2.3. Параметри, що однозначно визначають положення деталі S на площині

Для однозначного відображення положення деталі S на площині необхідно знати координати полюсу деталі (Xp^k, Yp^k) в системі координат *XOY*, що пов'язана із площиною, та кут повороту деталі відносно вихідного положення деталі θ_k (рис. 2.3).

Тоді координати будь-якої вершини $A_i^k(i=1,2...n)$ апроксимуючого многокутника для деталі S в системі координат *XOY*, що пов'язана із площиною, визначатимуться наступним чином[17-18]:

$$\begin{cases} Xm^k = Xm\cos\theta - Ym\sin\theta + Xp^k \\ Ym^k = Xm^i\sin\theta^k + Ym^i\cos\theta^k + Yp^k \end{cases} \qquad (2.10)$$

$i \quad i \quad k \quad i \quad k$

А координати будь-якої точки $Q^k(xq^k,yq^k)$ на стороні A_iA_{i+1} зовнішнього контуру апроксимуючого многокутника в системі координат *XOY*, що пов'язана із площиною, можна представити наступним чином:

$$\begin{cases} xq^k = (Xm_{i+1}^k - Xm_i^k)t_i + Xm_i \\ yq^k = (Ym_{i+1}^k - Ym_i^k)t_i + Ym_i \end{cases}, \quad \text{де } i=1,2...n-1 \text{ та } t_i \in [0,1]. \qquad (2.11)$$

$i+1 \quad i \quad i \quad i$

Тобто за допомогою виразів (2.10- 2.11) можна однозначно аналітично описати зовнішній контур деталі із заданою точністю ε у щільній укладці на площині.

2.3. Аналітичний опис умов взаємного неперетину деталей в укладці

Нехай S_1 та S_2 - два об'єкти, які зберігають постійну взаємну орієнтацію. Позначимо через O_1 та O_2 полюси об'єктів, які вибрані в довільних точках даних об'єктів. Тоді $X_1O_1Y_1$ та $X_2O_2Y_2$ - системи координат, які жорстко зв'язані з об'єктами S_1 та S_2 відповідно. Без обмеження відповідні координатні осі можна вважати направленими однаково. Припустимо, що об'єкт S_1 нерухомо закріплений на площині, а об'єкт S_2 - рухомий. Розглянемо множину можливих щільних положень об'єкта S_2 по відношенню до об'єкта S_1. Кожне таке положення характеризується вектором $\boldsymbol{r_{12}} = \boldsymbol{O_1O_2}$. Вектор- функція, що ставить у відповідність щільному положенню об'єктів S_1 та S_2

значення вектора $\boldsymbol{r_{12}}$ при умові, що об'єкт S_1 нерухомий, називається годографом вектор-функції Γ_{12} щільного розміщення (рис. 2.4) об'єкту S_2 відносно об'єкта S_1.

Нехай θ – кут, який утворює вектор $\boldsymbol{r_{12}}$ з віссю O_1X_1 . Тоді вектор-функція щільного розміщення об'єкта S_2 відносно об'єкта S_1 може бути задана у вигляді $\boldsymbol{r}= \boldsymbol{r}_{12}(\theta)$, де $0\leq \theta\leq 2\pi$[19] (див. рис. 2.4).

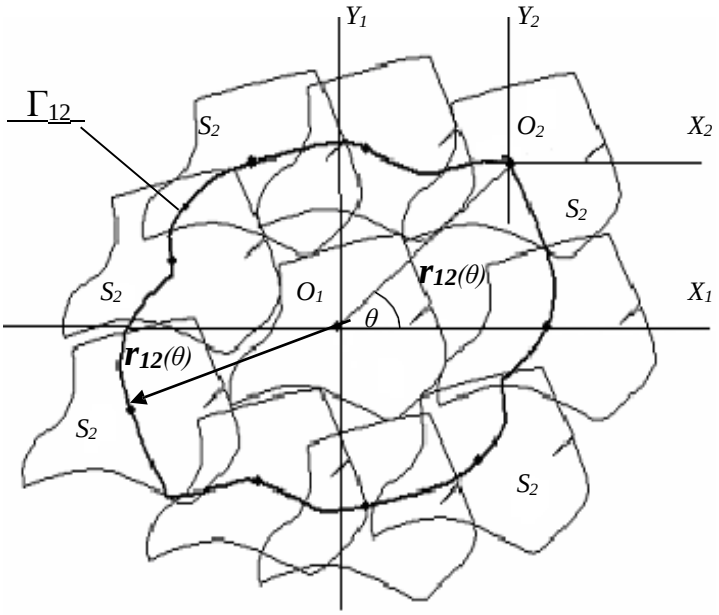

Рис. 2.4. Годограф вектор-функції щільного розміщення

Позначимо через Φ_{12} область, яка обмежена годографом Γ_{12}. Відмітимо найбільш важливі властивості годографа Γ_{12} вектор-функції щільного розміщення та області Φ_{12}, яка обмежена ним (рис. 2.5.):

- якщо полюс O_2 об'єкта S_2 розміщений всередині області Φ_{12}, то об'єкти S_1 та S_2 мають спільні внутрішні точки (перетинаються);
- якщо полюс O_2 об'єкта S_2 розміщений зовні області Φ_{12}, об'єкти S_1 та S_2 не мають спільних внутрішніх точок (не перетинаються);
- якщо полюс O_2 об'єкта S_2 розміщений на границі області Φ_{12}, то об'єкти S_1 та S_2 мають спільні граничні точки (щільно розміщені).

Годограф Γ_{12} двох однакових і однаково орієнтованих фігур має центральну симетрію. Крім того, годограф вектор-функції щільного розміщення

(ГВФЩР) об'єкту S_2 відносно об'єкта S_1 представляє собою замкнуту лінію та ГВФЩР для многокутників є також многокутник з числом вершин не більше ніж сума кількості вершин многокутників, для яких був побудований ГВФЩР.

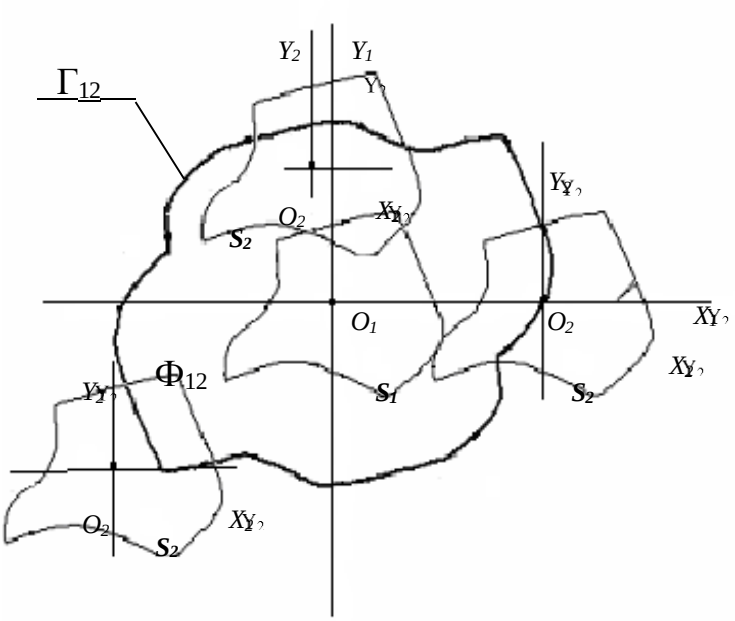

Рис. 2.5. Основні властивості годографа вектор-функції Γ_{12} щільного розміщення об'єкту S_2 відносно об'єкта S_1

Так як деталі взуття S_1 та S_2 ми апроксимуємо многокутниками S_{m1} та S_{m2}, то ГВФЩР для деталей взуття буде многокутник $\boldsymbol{G}$ з координатами вершин $G_i(Xg_i, Yg_i)$, $i=1,2..n_g$. Тоді координати будь-якої точки на ГВФЩР можна представити наступним чином:

$$\begin{cases} xg = (Xg_{i+1} - Xg_i)\tau_i - Xg_i \\ yg = (Yg_{i+1} - Yg_i)\tau_i - Yg_i \end{cases}, \text{ де } i=1,2..n_g \text{ та } \tau_i \in [0,1] \qquad (2.12)$$

Тобто, використовуючи апарат ГВФЩР ми маємо можливість контролювати взаємне розміщення деталей в укладці. Якщо полюси деталей будуть знаходитися на ГВФЩР, то деталі будуть дотикатись. Крім того апарат ГВФЩР дозволив за допомогою виразів (2.12) аналітично представити умови взаємного неперетину деталей, що розміщуються.

При побудові годографа в.-ф.щ.р. многокутника фактично реалізується процедура руху одного многокутника навколо іншого за умови, що в кожний момент руху контури, які їх обмежують, дотикаються.

Можливі три випадки[20] :

- вершина многокутника S_2 рухається по стороні многокутника S_1;
- сторона многокутника S_2 рухається по вершині многокутника S_1 ;
- сторона многокутника S_2 рухається по стороні мноокутника S_1 .

Оскільки всі внутрішні точки фігури при прямолінійно-поступальному русі без повороту рухаються однаково, то, прослідкувавши за траєкторією руху полюса, можемо описати рух всієї фігури. Фрагменти руху – це є рух полюса фігури по відрізках прямих. З цього випливає, що траєкторія руху полюса складається з відрізків, які послідовно з'єднані між собою. Початкова і кінцева точки траєкторії руху співпадають. Отже траекторія руху фігури, що є многокутником навколо іншого многокутника є замкненою ламаною лінією, тобто годограф Γ_{12} є замкнена ламана лінія.

Оскільки в кожний момент руху рухома і нерухома фігури дотикаються, то годограф Γ_{12} несе в собі інформацію про взаємне розташування фігур, при якому досягається їх щільне розміщення.

Розглянемо схему обчислення координат вершин годографа двох опуклих многокутників.

Зафіксуємо на площині фігуру S_1. Розташуємо фігуру S_2 так, щоб вона дотикалася до S_1. З кожною з цих фігур зв'яжемо їх власні системи координат: $X_1O_1Y_1$ з S_1 та $X_2O_2Y_2$ з S_2. Інформацію про кожну з фігур задамо у вигляді послідовності координат вершин у власних системах координат. Позначимо послідовність вершин фігури S_1 як $A'_i(x'_i, y'_i)$, $i = \overline{1,n}$, а фігури S_2 - $A'_j(x'_j, y'_j)$, $j = \overline{1,n}$.

В будь-який момент руху фігури S_2, відповідні координатні осі координатних систем $X_1O_1Y_1$ та $X_2O_2Y_2$ мають одинаковий напрямок. Будемо вважати, що координатна система $X_1O_1Y_1$, яка звязана з S_1, співпадає з координатною системою XOY на площині і в ній обчислимо координати вершин годографа.

Початком траєкторії руху будемо вважати взаємне розташування фігур, при якому деяка вершина $A'_i(x'_i, y'_i)$ фігури S_1 суміщена з деякою вершиною $A'_j(x'_j, y'_j)$ фігури S_2 і фігури не перетинаються. Такому розміщенню фігур відповідає вершина $G_k(x_k; y_k)$ годографа Γ_{12}:

$$G_k(x_k, y_k): \quad \begin{cases} x_k = x'_i - x'_j \\ y_k = y'_i - y'_j \end{cases} \tag{2.13}$$

Для того, щоб обчислити координати наступної вершини годографа необхідно визначити напрямок переміщення фігури S_2 відносно фігури S_1.

Щоб зберегти щільне розміщення фігур на наступному етапі повинна реалізуватися одна з двох можливостей:

- або вершина A'_j фігури S_2 буде рухатися по прилеглій стороні $A'_i A'_{i+1}$ нерухомої фігури S_1;
- або прилегла сторона $A'_j A'_{j+1}$ фігури S_2 повинна рухатися по вершині A'_i нерухомої фігури S_1.

Основним при виборі напрямку руху є забезпечення умови, щоб в процесі руху внутрішня частина фігури S_2 не накладалася на внутрішню частину нерухомої фігури S_1, тобто щоб фігури не перетиналися.

При обраній орієнтації контурів фігур внутрішня частина кожної з фігур прилягає до її контура з лівого боку. Необхідно обрати такий напрямок руху, при якому рухома фігура буде знаходитися з іншого боку від контура нерухомої

фігури на ділянці дотику фігур. Коли вершина $A'_i(x_i', y_i')$ фігури S_1 суміщена з вершиною $A'_j'\left(x'_j, y'_j\right)$ фігури S_2, то подальший рух фігури S_2 буде відбуватися:

- або вздовж прямої на якій лежить сторона $A_i'A_{i+1}'$ фігури S_1;
- або вздовж прямої, на якій лежить сторона $A'_j'A'_{j+1}'$ фігури S_2.

Щоб при цьому фігури S_1 і S_2 не перетиналися, слід обрати ту з прямих, для якої всі вершини фігури S_1 знаходяться з лівого боку від прямої, а всі вершини фігури S_2 - з правого боку від неї.

Введемо спеціальну функцію Z_{ij}

$$Z_{ij} = \begin{vmatrix} x_{i+1}' - x_i' & y_{i+1}' - y_i' \\ x'_{j+1} - x'_j & y'_{j+1} - y'_j \end{vmatrix}, \qquad (2.14)$$

за знаком якої можна встановити напрямок руху фігури S_2.Обчисливши її значення при відповідних i і j, що відповідають індексам суміщених вершин, отримаємо один з випадків:

- якщо $Z_{ij} > 0$, то напрямок руху фігури S_2 вказує вектор $\overrightarrow{A'_j'A'_{j+1}'}$;
- якщо $Z_{ij} < 0$, то напрямок руху фігури S_2, вказує вектор $\overrightarrow{A'_iA'_{+i\ 1}}$;
- якщо $Z_{ij} = 0$, то маємо випадок, коли вектори $\overrightarrow{A'_j'A'_{j+1}'}$ і $\overrightarrow{A'_iA'_{+i\ 1}}$ колінеарні і напрямок руху фігури S_2 визначається вектором $\overrightarrow{A'_iA'_{+i\ 1}}$.

На кожному кроці координати вершин годографа обчислюються за формулами (67).

В загальному випадку не має значення, які дві вершини ми сумістимо на першому етапі обчислень. Тому за початкове оберемо таке розташування фігур S_1 і S_2, при якому вершина фігури S_1 з максимальною абсцисою співпадає з вершиною фігури S_2 з мінімальною абсцисою.

Обчислення закінчуються, коли годограф Γ_{12} “замкнеться”, тобто координати чергової знайденої вершини годографа співпадуть з координатами його першої вершини. В результаті обчислень отримуємо масив $G_k(x_k; y_k)$ $k=\overline{1,m}$ координат послідовних вершин годографа Γ_{12} фігур S_1 і S_2.

Записавши в параметричному вигляді рівняння відрізків, що є ребрами годографа, отримаємо аналітичне описання годографа Γ_{12} у вигляді системи співвідношень:

$$\begin{cases} x(\tau_i) = x + (x_{i+1} - x_i)\tau_i \\ y(\tau_i) = y + (y_{i+1} - y_i)\tau_i \end{cases} \quad i=\overline{1,m} \qquad (2.15)$$

де $(x_i; y_i)$ координати вершин годографа Γ_{12}, τ_i - параметр, $\tau_i \in [0;1)$.

2.4. Алгоритм побудови опуклої оболонки для многокутника

Постановка задачі. . Нехай маємо многокутника ***P*** з координатами вершин (Xp_i, Yp_i), *i=0,1,2..n-1*. Знайти опуклу оболонку ***R*** із координатами вершин (Xob_j,Yob_j), *j=0,1,2..p-1* для многокутника ***P***.

Алгоритм побудови опуклої оболонки буде базуватись на властивості векторного добутку двох векторів. ***a***=(*xa*,*ya*) та ***b***=(*xb*,*yb*), а саме: модуль векторного добутку буде додатнім, якщо найкоротший шлях від вектора ***a*** до вектора ***b*** буде проти годинникової стрілки.

На рис. 2.5 представлені етапи побудови опуклої оболонки для многокутника та опорної функції. Спочатку вибираємо за початкову вершину. в якій координата *X* досягає максимуму. Це буде точка P_0 (рис.2.5.а). Потім будуємо вектор ***a***=P_0P_1 ***b***=P_0P_2 . Якщо модуль векторного добутку двох векторів. ***a*** та ***b*** буде менше або дорівнювати нуля при обході контуру проти годинникової стрілки, то вершина P_1 не знаходиться на опуклій оболонці і буде виключена із розгляду (рис. 2.5.б). Аналогічно продовжуємо розгляд контуру многокутника та вилучення з розгляду тих вершин многокутника, які не лежать на опуклій

оболонці (рис. 2.5.в) поки не залишиться жодної точки на новому контурі, яка не належить опуклій оболонці (рис. 2.5.в-г). Побудована опукла оболонка та опорна функція для опуклої оболонки представлені на рис.2.5.г[21-23].

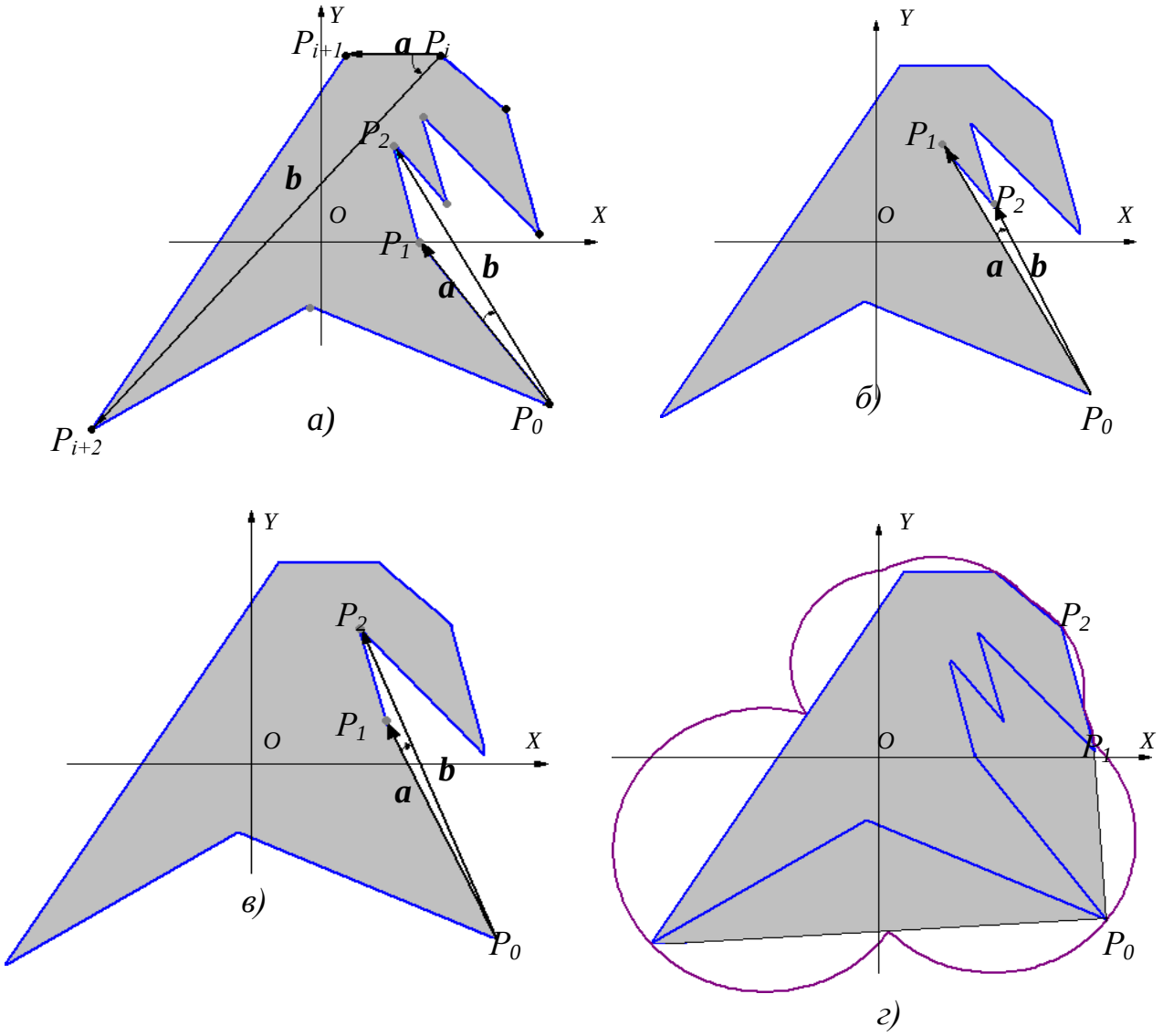

Рис.6.21. Етапи побудови опуклої оболонки та опорнох функції

Алгоритм побудови опуклої оболонки ***R*** для многокутника ***P*** матиме наступні пункти:

1. Визначення на многокутнику такої вершини, що вона завжди входить до вершин на зовнішньому контурі опуклої оболонки. За таку вершину візьмемо

будь-яку з вершин многокутника, в якій координата *X* або *Y* досягає екстремуму (максимуму або мінімуму). Нехай за таку вершину приймемо *к* -ту вершину многокутника, для якої $Xp_k = \max_{i=0,1..n-1}(Xp_i)$;

2. В многокутнику ***P*** циклічно переставимо вершини таким чином, що першою вершиною стане *к*-та вершина многокутника, для якої $Xp_k = \max_{i=0,1..n-1}(Xp_i)$;

3. Визначаємо напрям обходу многокутника (порядок, у якому записані координати вершин многокутника при обході многокутника, за годинниковою стрілкою чи проти годинникової стрілки). Для цього знаходимо порядкові номери вершин *k1, k2, k3,* для яких виконуються наступні вирази

$$Yp_{k1} = \max_{i=0,1..n-1}(Yp_i); \quad Xp_{k2} = \min_{i=0,1..n-1}(Xp_i); \; Yp_{k3} = \min_{i=0,1..n-1}(Yp_i).$$

Якщо виконується наступна нерівність *k1≤ k2≤ k3,* то напрям обходу контуру многокутника проти годинникової стрілки, інакше напрям обходу контуру многокутника за годинниковою стрілкою.

5. Якщо напрям обходу контуру многокутника проти годинникової стрілки, то *Flag1=True,* інакше - *Flag1=False.*

6. Присвоюємо $i=0,\ j=0,\ t=1,\ m=2,\ Flag2=True,\ Xob_0=Xp_0,\ Yob_0=Yp_0$;

7. Визначимо $\Delta = (Yp_{i+m} - Yp_i)(Xp_{i+t} - Xp_i) - (Xp_{i+m} - Xp_i)(Yp_{i+t} - Yp_i)$.

Якщо $\Delta<0$ та *Flag1=True* або $\Delta>0$ та *Flag1=False,*

то виконуємо наступні дії

Початок *Flag1=False; q=i+m; m=m+1; l=l+1* ***Кінець***

інакше виконуємо наступні дії

Початок $j=j+1;\ q=i+m+1;\ m=m+1;\ Xob_j=Xp_{i+l},\ Yob_j=Yp_{i+l};$

$i=i+l;\ m=2;\ l=1$ ***Кінець***

8. Якщо $q<n-1$, то повертаємося до пункту 7;

9. Присвоюємо $Xob_{j+1}=Xp_0$, $Yob_{j+1}=Yp_0$, $p=j+2;$

10. Якщо *Flag2=False,* то

Початок для $j=0$ до $p-1$ роби

Початок $Xp_j = Xob_j$, $Xp_j = Xob_j$ ***Кінець***

$m=p;$ Повертаємося до пункту 6 , ***Кінець***

інакше ***Кінець***.

Висновки до другого розділу

Запропонована математична модель задачі побудови щільних укладок плоских об’єктів, близьких до опуклих, виділені її структурні компоненти та представлений аналітичний опис цих компонентів. Це дозволило розробити алгортими вирішення задачі побудови щільних укладок плоских об’єктів, близьких до опуклих.

3.РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРОЄКТУВАННЯ ДЕКОРАТИВНИХ ЕЛЕМЕНТІВ НА ДЕТАЛЯХ ВЗУТТЯ

3.1. Вимоги до програмного продукту

Програмний продукт повинен забезпечувати:

- 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. Програмні засоби

Для реалізації даного програмного продукту необхідні наступні програмні засоби: Delphi 7.0.

Цей вибір обґрунтовано тим, що використання даних інструментальних засобів дозволяє створити інформаційну систему для обробки і аналізу необхідного взаємопов’язаного набору даних[23].

Як операційна система була обрана ітерактивна оболонка Microsoft Windows – 2010, що дозволяє підтримувати режим багатозадачності, а також на відзнаку від відомих операційних систем Unix, OS/2 та ін. Windows забезпечує високу якість зображення. Вона володіє розширеними можливостями при паралельній роботі декількох програм, спрощеною передачею результатів від програми до програми, підвищеною ємністю системи, дозволяє водночас виконувати велике число прикладних програм і системних компонентів, ефективними засобами управління оперативною пам'яттю і іншими можливостями, що відрізняють її від вищезгаданих операційних систем.

Delphi 7.0 – це програмний інструмент для створення прикладних програм, які працюють в середовищі операційної системи Microsoft Windows – 2010. Пакет включає не тільки мову програмування, а й ефективне діалогове середовище для розробки форм та вікон, що дозволяє програмісту створити програмний продукт достатньо високої якості за порівняно невеликий проміжок часу. Delphi 7.0 дає можливість використовувати ресурси інших прикладних програм Windows та взаємодіяти з зовнішньою базою даних[24].

3.3. Опис основних процедур розробленого програмного продукту, які забезпечують побудову щільних укладок для плоских об’єктів, близьких до опуклих

Процедура *TForm1.Open1Click* забезпечує введення інформації про зовнішні контури деталей моделі взуття для нанесення декоративних елементів на них[25-26].

```
procedure TForm1.Open1Click(Sender: TObject);

  procedure vvod;
  var f:system.text;
  filename:string;
  xi,yi,xe,ye,xmax,ymax,xmin,ymin:int64;
  i,j:word;
  nxi:integer;
  xrab,yrab:real;
  A:string;
  begin
  if opendialog1.execute then
  begin
  filename:=opendialog1.filename;
  system.Assign(f,filename);
  reset(f);
  readln(f,model);
  caption:='îîääåëü'+model;
  readln(f,a);
  readln(f,koldet);
  for i:=1 to koldet do
  readln(f,namedet[i]);
```

```
for i:=1 to koldet do
 begin
 readln(f,kolpointdet[i]);
 nn[i]:=kolpointdet[i];
 end;
for i:=1 to koldet do
 begin
 for j:=0 to nn[i]-1 do
 begin
 readln(f,xrab,yrab);
 if kolpointdet[i]<>nn[i] then continue;
 if (xrab=999)and (yrab=999) then
 begin
  kolpointdet[i]:=j;
  continue
  end;
  xm[i,j]:=round(xrab*100);
  ym[i,j]:=round(yrab*100);

      end;
       nn[i]:=kolpointdet[i];
         end;
      for i:=1 to koldet do begin

  maxmin(nxi,kolpointdet[i],xm[i],xi,-1);
  maxmin(nxi,kolpointdet[i],ym[i],yi,-1);
  maxmin(nxi,kolpointdet[i],xm[i],xe,1);
  maxmin(nxi,kolpointdet[i],ym[i],ye,1);
```

```
dl_det[i]:=xe-xi;
sh_det[i]:=ye-yi;
if  i=1  then  begin
dl_dt:=dl_det[i];
sh_dt:=sh_det[i];
end
 else
 begin
   if dl_dt<dl_det[i] then dl_dt:=dl_det[i];
   if sh_dt<sh_det[i] then sh_dt:=sh_det[i];
 end;

if i=1 then
 begin
 xmax:=xe;
 ymax:=ye;
 xmin:=xi;
 ymin:=yi;
 xcdet[1]:=round((xe+xi)/2);
 ycdet[1]:=round((ye+yi)/2);

 end
 else
 begin
 xcdet[I]:=round((xe+xi)/2);
 ycdet[I]:=round((ye+yi)/2);
// DlDet[i]:=Xe-Xi;
// ShDet[i]:=Ye-Ye;
```

```
        if xmax<xe then xmax:=xe;
        if ymax<ye then ymax:=ye;
        if xmin>xi then xmin:=xi;
        if ymin>yi then ymin:=yi;
        end
          end;  end;
          xc:=round((xmax+xmin)/2);
          yc:=round((ymax+ymin)/2);
          dl:=xmax-xmin;
          sh:=ymax-ymin;
      for i:=1 to koldet do
      begin
        for j:=0 to kolpointdet[i]-1 do
         begin
         xm[i,j]:=xm[i,j]-xcdet[i];
         ym[i,j]:=ym[i,j]-ycdet[i];
         end;
{         xcdet[i]:=xcdet[i]+xc;
         ycdet[i]:=ycdet[i]+yc;}
         SCALE(xmax,xmin,ymax,ymin,image1.Width,image1.height, mxy);

         Obxod(KolPointDet[i],xm[i],ym[i]);
        end ;
         system.Close(f);
       end;

    begin
    TabSheet1.Visible:=True;
```

```
   TabSheet2.Visible:=True;
   TabSheet3.Visible:=True;
   TabSheet4.Visible:=True;
   TabSheet1.TabVisible:=True;
   TabSheet2.TabVisible:=True;
   TabSheet3.TabVisible:=True;
   TabSheet4.TabVisible:=True;
   vvod;
   ScrollBox1.Visible:=true;
     With form1.Image2 .Canvas  do
begin
 pen.Color :=clred;
 form1.Image2 .Height :=koldet*55+10;
 scale_image2(mmxy);
 for i:=1 to koldet do  begin
 XcYc(kolpointdet[i],xm[i],ym[i],xc,yc);

Rectangle(5,5+55*(i-1),55,50+55*(i-1))  ;
Graph_im2(i,mmxy,kolpointdet[i],  xm[i],ym[i], 27,round(27+55*(i-1)),xc,yc);
end;
end;

end;

procedure scale_im2(var mx:real);
var   my:real ;
begin
mx:=form1.Image2.Width/dl_dt ;
```

```
my:=form1.Image2.Height /sh_dt ;
if mx>my then mx:=my;

end;
```

Процедура *Graph_im1* забезпечує вивід зображення многокутника на область *Image1*[27-28].

Параметри процедури:

- *col*-номер кольору:

```
1:  pen.Color:=ClRed;
2:  pen.Color:=ClBlack;
3:  pen.Color:=ClGray;
4:  pen.Color:=ClYellow;
5:  pen.Color:=ClPurple;
6:  pen.Color:=ClSilver;
7:  pen.Color:=ClGreen;
8:  pen.Color:=ClAqua;
9:  pen.Color:=ClBlue;
10:  pen.Color:=ClFuchsia;
11:  pen.Color:=ClTeal;
12:  pen.Color:=ClNavy;
13:  pen.Color:=ClMaroon;
14:  pen.Color:=ClOlive;
```

mmxy - масштаб, в якому виводиться зображення многокутника вобласть *Image1;*

n - *кількість* вершин многокутника*;*

xm,ym - масиви з координатами вершин многокутника*;*

xc,yc – координати центру прямокутника, описаного навколо многокутника, зображення якого виводиться в область *Image1;*

xcd,ycd – координати центру області *Image1;*

```
procedure Graph_im1(col:integer;mmxy:real; n:word; xm,ym:array of int64;
                    xc,yc:integer; xcd,ycd:int64);
var Xr,Yr:mas1; i:Integer ;
```

```
Begin
for i:=0 to n-1
begin
xr[i]:=round((xm[i]-xcd)*mmxy+xc);
yr[i]:=round((-ym[i]+ycd)*mmxy+yc);
end;
with form1.image1.canvas do  begin
Pen.width:=2;
Pen.Mode:=pmCopy;
case col of
1:  pen.Color:=ClRed;
2:  pen.Color:=ClBlack;
3:  pen.Color:=ClGray;
4:  pen.Color:=ClYellow;
5:  pen.Color:=ClPurple;
6:  pen.Color:=ClSilver;
7:  pen.Color:=ClGreen;
8:  pen.Color:=ClAqua;
9:  pen.Color:=ClBlue;
10:  pen.Color:=ClFuchsia;
11:  pen.Color:=ClTeal;
12:  pen.Color:=ClNavy;
13:  pen.Color:=ClMaroon;
14:  pen.Color:=ClOlive
else pen.Color:=TColor(RGB(255,128,64));
end;
for i:=0 to n-1 do
if i=0 then moveto(xr[i],yr[i]) else lineto(xr[i],yr[i]); {Ðèñóåì êîíòóðû äåòàëè}
end;
End;
```

Процедура *Skv* визначає площу *s* многокутника, який визначається наступними параметрами:

n - *кількість* вершин многокутника*;*

x,y - масиви з координатами вершин многокутника.

```
Procedure Skv(n:integer; Var x,y:array of Int64; Var s:Int64);
Var i:integer;
Begin
```

```
s:=0;
For i:=0 to n-2 do
s:=s+x[i]*y[i+1]-x[i+1]*y[i];
s:=Round(abs(s)/2)
End;
```

Процедура *PrKoor* забезпечує перерахування координат вершин многокутника при повороті його на кут *alf*, який визначається наступними параметрами:

n - *кількість* вершин многокутника*;*

x,y - масиви з координатами вершин многокутника;

xr,yr - масиви з координатами вершин многокутника при повороті його на кут *alf*.

```
Procedure PrKoor(n:integer; var x,y,xr,yr:array of Int64;
alf:real;xr1,yr1:Int64);
Var i:integer;
begin
for i:=0 to n-1 do
begin
xr[i]:=round(x[i]*cos(alf)-y[i]*sin(alf)+xr1);
yr[i]:=round(x[i]*sin(alf)+y[i]*cos(alf)+yr1);
end;
end;
```

Процедура *VipDet* генерує опуклий мнокутник(опуклу оболонку) з координатами вершин *Xob,Yob* навколо многокутника довільної форми з координатами вершин *X, Y*.

Параметри процедури:

n - кількість вершин базового многокутника *;*

X,Y - масиви з координатами вершин базового многокутника;

Nob - кількість вершин опуклого мнокутника(опуклої оболонки) навколо многокутника довільної форми з координатами вершин *X, Y*.

Xob,Yob – координати вершин опуклого мнокутника(опуклої оболонки) навколо многокутника довільної форми з координатами вершин *X, Y*.

```
Procedure VipDet(n:integer; x,y:array of Int64; Var Xob,Yob:array of Int64;
                 Var Nob:integer);
  Var
   i,l,m,p,q:integer;
     delta:real;
     pr:boolean;
  Begin
    p:=0;
    Xob[p]:=X[0];
    Yob[p]:=Y[0];
    Nob:=n;
   Repeat
   i:=0;
    m:=2; l:=1;
    pr:=true;
    Repeat
     delta:=(y[i+m]-y[i])*(x[i+L]-x[i])-
           (y[i+L]-y[i])*(x[i+m]-x[i]);
     if delta<=0 then
       begin
        pr:=false;
        q:=i+m;
        m:=m+1;
        L:=L+1;
//         q:=i+m;
       end
```

```
        else
         begin
         p:=p+1;
          q:=i+m+1;
          Xob[p]:=X[i+L];
          Yob[p]:=Y[i+L];
          i:=i+L;
          m:=2;
          L:=1
         end;
        Until q>Nob-1;
        Xob[p+1]:=X[0];
        Yob[p+1]:=Y[0];
        Nob:=p+2;
        p:=0;
        For i:=0 to Nob-1 do
         begin
          X[i]:=Xob[i];
          Y[i]:=Yob[i];
         end;
      Until pr;
    End;
```

Процедура *GodVip* генерує годограф вектор-функціх щільного розміщення для двох опуклих мнокутників.

Параметри процедури:

nn - кількість вершин нерухомого многокутника *;*

Xn,Yn - масиви з координатами вершин нерухомого многокутника;

nn - кількість вершин рухомого многокутника *;*

Xn,Yn - масиви з координатами вершин рухомого многокутника;

k - кількість вершин годограф вектор-функціх щільного розміщення для двох опуклих мнокутників.

Xg,Yg –координати вершин годографа вектор-функціх щільного розміщення для двох опуклих мнокутників.

```
Procedure GodVip(nn,nv:integer; Var xn,yn,xv,yv,
                 xg,yg:array of Int64; Var k:integer);
  Var i,j:integer;
    s:real;
  begin
{   writeln(nn:4,nv:4,xn[1]:5,yn[1]:5,xv[1]:5,yv[1]:5);}
   i:=0;
   j:=0;
   k:=-1;
   xn[nn]:=xn[1];
   yn[nn]:=yn[1];
   xv[nv]:=xv[1];
   yv[nv]:=yv[1];
   xn[nn+1]:=xn[2];
   yn[nn+1]:=yn[2];
   xv[nv+1]:=xv[2];
   yv[nv+1]:=yv[2];
   repeat
    k:=k+1;
    xg[k]:=xv[i]-xn[j];
    yg[k]:=yv[i]-yn[j];
    s:=(xn[j]-xn[j+1])*(yv[i+1]-yv[i])-
```

```
                (yn[j]-yn[j+1])*(xv[i+1]-xv[i]);
                  if s<=0 then i:=i+1
                  else j:=j+1;
{                writeln(k:3,xg[k]:7:2,yg[k]:7:2);
               readln;                    }
              until ((k>nn-1) and (xg[0]=xg[k]) and
                  (k>nv-1) and (yg[0]=yg[k]));
                  k:=k+1
           end;
```

Функція *PointCrossTwoLine* повертає *True* та координати вершин точки перетину двох відрізків, якщо вони перетинаються[29-30]. Інакше вона повертає *False.*

Параметри функції:

Xa,Ya,Xb,Yb (Xc,Yc,Xd,Yd)- відповідно координати вершин першого (другого) відрізку.

x0,y0 – координати точки перетину двох відрізків.

```
Function PointCrossTwoLine(Xa,Ya,Xb,Yb,Xc,Yc,Xd,Yd:Int64; var x0,y0:Int64):boolean;
  Var  A1,B1,C1,A2,B2,C2,Ra,Rb,Rc,Rd,Rab,Rcd,D1,D2,D0:Int64;
     i:integer;
     B:boolean;
  Begin
    PointCrossTwoLine:=False;
    A1:=ya-yb;
    B1:=xb-xa;
    C1:=yb*(xa-xb)-xb*(ya-yb);
    A2:=yc-yd;
```

```
        B2:=xd-xc;
        C2:=yd*(xc-xd)-xd*(yc-yd);
        Ra:=A2*Xa+B2*Ya+C2;
        Rb:=A2*Xb+B2*Yb+C2;
        Rc:=A1*Xc+B1*Yc+C1;
        Rd:=A1*Xd+B1*Yd+C1;
        Rab:=Ra*Rb;
        Rcd:=Rc*Rd;
B:=((Rcd<0)and(Rab<0))or((Rcd<0)and(Rab=0))or((Rcd=0)and(Rab<0));
        If B then
         begin
          PointCrossTwoLine:=True;
          D0:=A1*B2-A2*B1;
          D1:=C2*B1-C1*b2;
          D2:=C1*A2-C2*A1;
          X0:=Round(D1/D0);
          Y0:=Round(D2/D0);
         end;
       End;
```

3.4. Інструкції по роботі з програмним продуктом

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

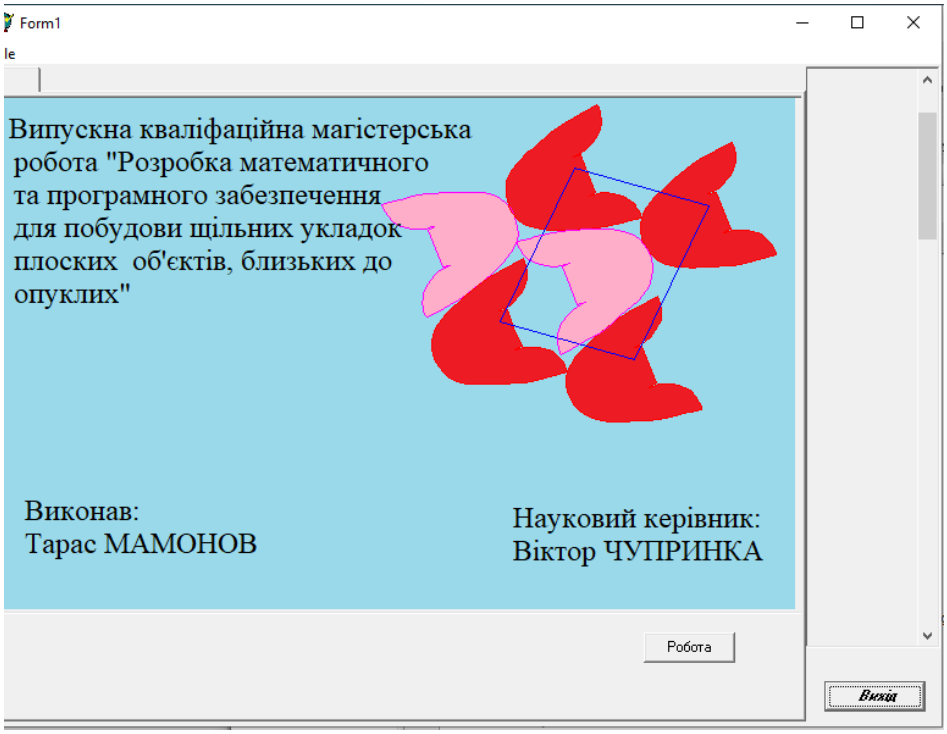

Рис. 3.1

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

,

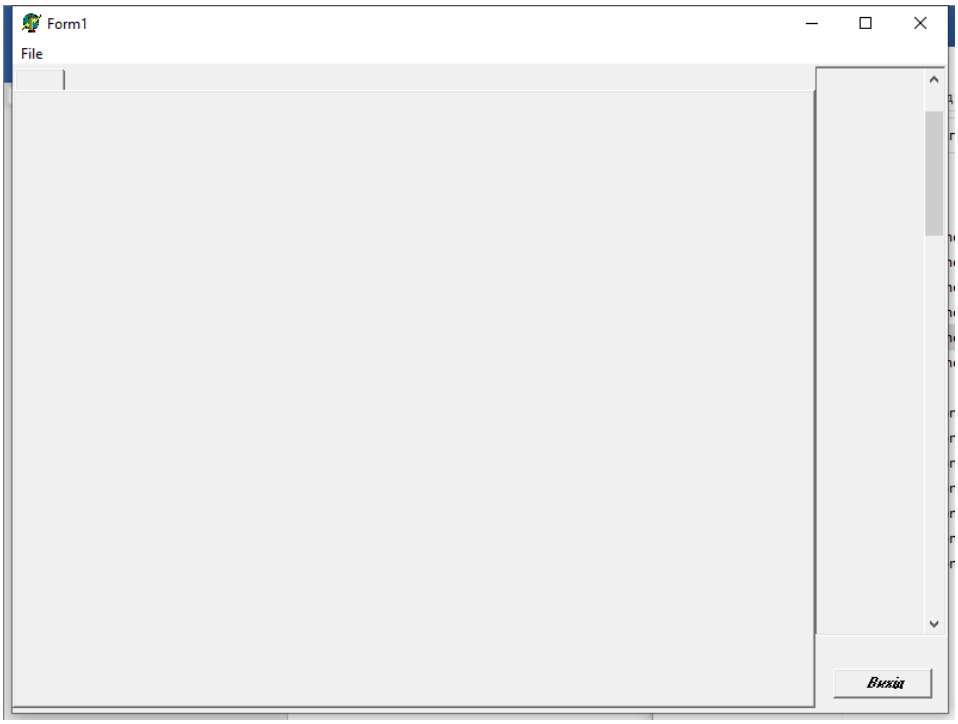

Рис. 3.2

Тепер вводимо інформацію про зовнішні деталей моделі. Після цього головне вікно програми настуний вигляд (рис. 3.3).

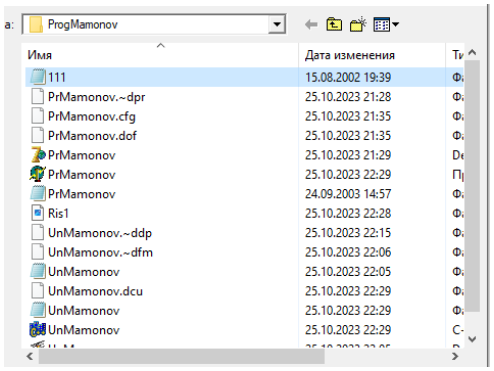

Рис. 3.3

Після вводимо інформацію про зовнішні деталей моделі головне вікно програми настуний вигляд (рис. 3.4).

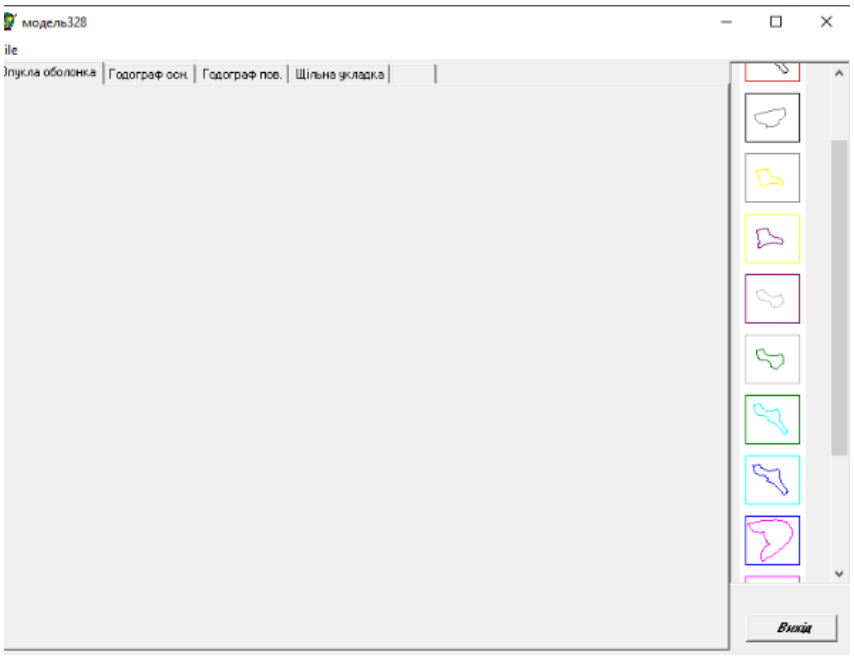

Рис. 3.4

Після вибору деталі із меню, що знаходиться праворуч на формі, головне вікно програми настуний вигляд (будується опукла оболонка деталі)(рис. 3.5).

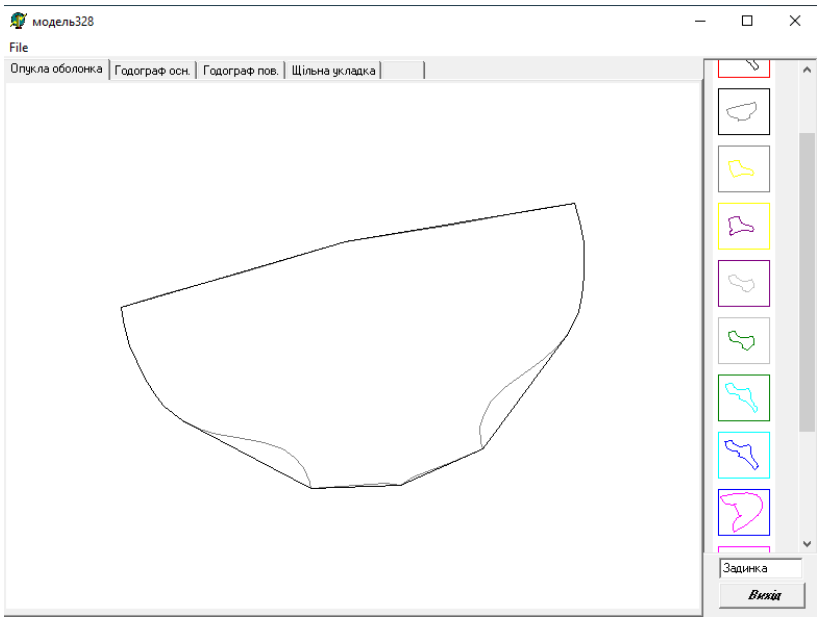

Рис. 3.5

Після побудови опуклої оболонки деталі будується годограф вектор-функції щільного розміщення для деталі самої з самою (рис. 3.6).

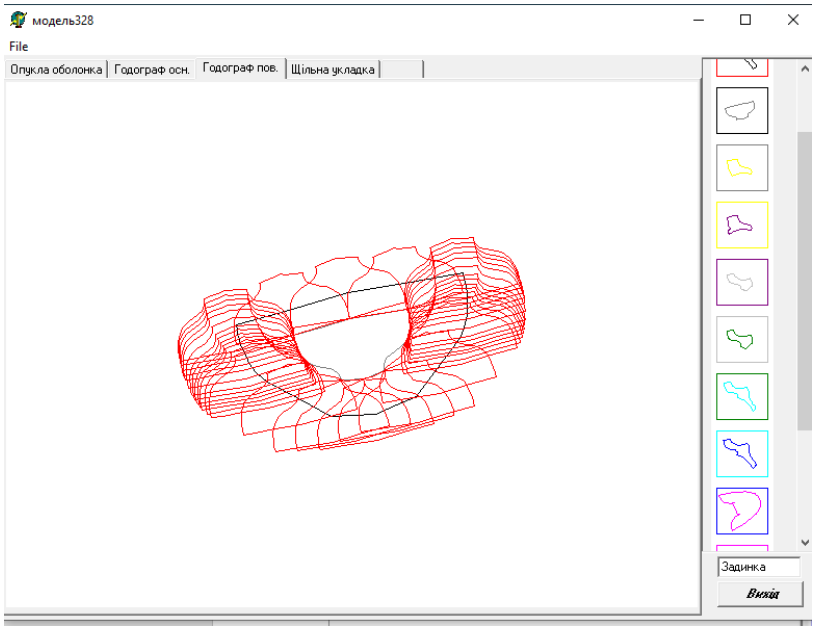

Рис. 3.6

Далі будується щільна укладка для для однаково орієнтованих деталей(рис. 3.7).

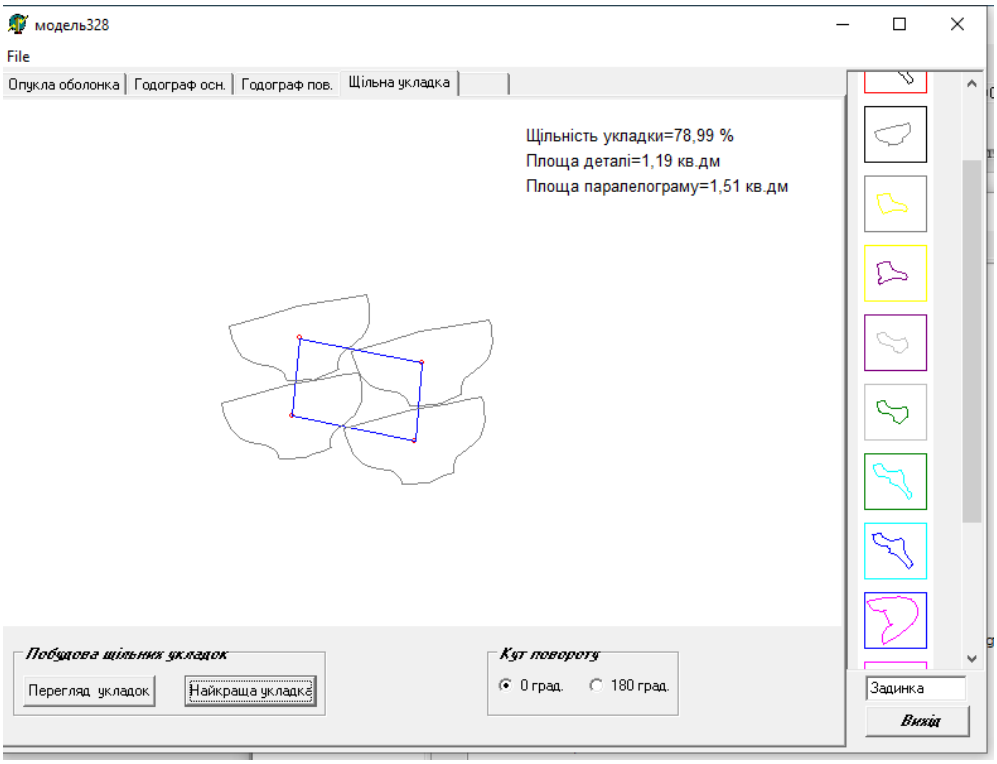

Рис. 3.7

Програма має молжливість щільна укладка для деталей, що повернуті на 180^0(рис. 3.8)

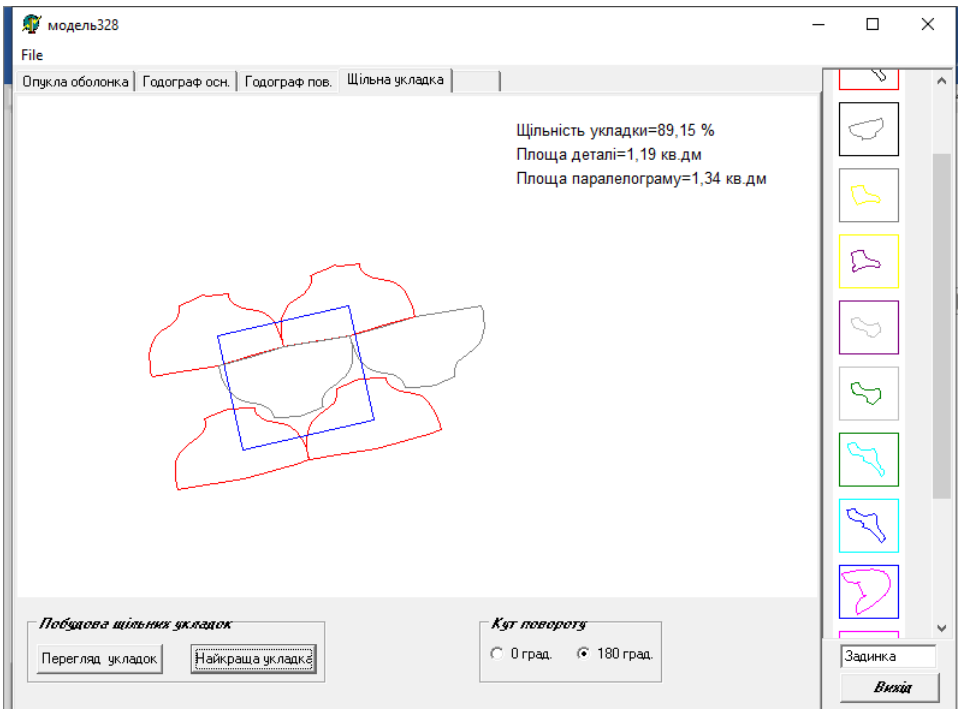

Рис. 3.8

Висновки до третього розділу

Запропонована математична модель задачі побудови щільних укладок плоских об’єктів, близьких до опуклих, виділені її структурні компоненти та представлений аналітичний опис цих компонентів та запропоновані алгортими вирішення задачі побудови щільних укладок плоских об’єктів, близьких до опуклих, дозволили розробити програмний продукт, який вирішує поставлену задачу.

Висновки

Проведено дослідження та розроблені такі алгоритми для автоматизованого проектування щільних укладок для плооских геометричних обєктів, що є близькі до опуклих:

- Побудова опуклої оболонки для плооских геометричних обєктів, що є близькі до опуклих;
- Генеруванн щільних укладок для однаково орієнтованих плооских геометричних обєктів, що є близькі до опуклих;
- Генеруванн щільних укладок для повернутих на 180^0 плооских геометричних обєктів, що є близькі до опуклих;
- Запропоноване математичне забезпечення для генерування щільних укладок для однаково орієнтованих плооских геометричних обєктів, що є близькі до опуклих;

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Чертенко Л.П. Розробка комп'ютерної технології проектування внутрішньої форми взуття та деталей низу автореф. дис. на здобуття наук. ступеня канд. техн. наук : 05.16.08. Київ, 2003. 27 с.
2. Чертенко Л.П., Коновал В.П. Математичне задання контурів внутрішньої форми взуття // Вісник КНУТД. 2002. No1. С. 15-19.
3. Каменець С.Є., Васютинська В.В. Використання сучасних методів візуального дизайну та просторового моделювання для створення шкіргалантерейних виробів// Вісник Хмельницького національного університету. – 2020. - № 4. с. 199-205.
4. Gerber Hiera mit Partner//Schuh-Techn. Int. Schuh-Techn. + ABC. 1997.- 91, № 1 -2.- P. 6
5. Lectra-Marktfiihrer in Brasilien//Schuh-Techn. Int.1995.-89, № 1 - 2. P.24-25.
6. Практикум з конструювання і проектування взуття: навч. посібник / за заг. ред. В. І. Бегняк. – Хмельницький: ХНУ, 2013. – С. 173–180. Каменець С.Є., Васютинська В.В. Використання сучасних методів візуального дизайну та просторового моделювання для створення шкіргалантерейних виробів// Вісник Хмельницького національного університету. – 2020. - № 4. с. 199-205.
7. Коновал В. П. Універсальний довідник взуттєвика: навчальний посібник / В. П. Коновал, С. С. Гаркавенко, Л. Т. Свістунова. – Київ: Лібра, 2005. – 720 с.
8. Столярова В. В., Каменець С. Є., Борщевська Н. М. Просторове моделювання та проектування аксесуарів і фурнітури виробів індустрії моди / Технології та дизайн № 3(36) 2020р. [Електронний ресурс] - Режим доступу:https://drive.google.com/file/d/1DEkBLCjWLsRG23OkK0hgnKNDO-uZTZ6i/ view?usp=sharing. Gerber Hiera mit Partner//Schuh-Techn. Int. Schuh-Techn. + ABC. 1997.- 91, № 1 -2.-P. 6

9. Чупринка В.І. Підготовка інформації для автоматичного розкрою / В.І. Чупринка, Волошин, Піпа Т.А. // Вісник ДАЛПУ. – 2000. - №1. – С. 91-83
10. Чупринка В.І. Розрахунок площі деталей, зовнішні контури яких апроксимуються В-сплайном / В.І. Чупринка, П.В. Омельченко // Вісник ДАЛПУ. - 2000. - № 1. – С. 89-90.
11. Чупринка В.І. Проектування раціональних схем розкрою рулонних матеріалів на деталі взуття з урахуванням розмірного асортименту / В.І. Чупринка, О.О. Шишкіна, Л.Т. Свістунова // Вісник КНУТД - 2002. - №2. – С. 35-38
12. Програмний комплекс для проектування раціональних схем розкрою рулонних матеріалів на деталі взуття з урахуванням розмірного асортименту. / В.І. Чупринка, О.О. Шишкіна, Л.Т. Свістунова, Т.І. Тимчук // Вісник Технологічного університету Поділля. - 2003. - №4, Ч2. – С. 110-113..
13. 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
14. 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.
15. Чупринка В.І. Алгоритм автоматичної підготовки вихідної інформації для побудови раціональних схем розкрою. / В.І. Чупринка, О.В. Чебанюк // Вісник КНУТД. – 2006. - №6. – С. 18-22
16. Чупринка В.І. Підготовка інформації для автоматичного розкрою / В.І. Чупринка, Волошин, Піпа Т.А. // Вісник ДАЛПУ. – 2000. - №1. – С. 91-83.
17. 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

18. 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.
19. Стоян Ю.Г. Розміщення геометричних об’єктів. – Київ: Наукова думка, 1975. – 239с.
20. Стоян Ю.Г., Гиль І.М. Методи та алгоритми розміщення плоских геометричних об’єктів. – Київ: Наукова думка, 1976. – 242с.
21. Чупринка В.І. Розрахунок площі деталей, зовнішні контури яких апроксимуються В-сплайном / В.І. Чупринка, П.В. Омельченко // Вісник ДАЛПУ. - 2000. - № 1. – С. 89-90
22. Чупринка В.І. Алгоритм побудови щільних укладок для двох видів плоских геометричних об’єктів. / В.І. Чупринка, О.В. Чебанюк // Вісник КНУТД. – 2007. -№6. – С. 107-112
23. .Ковалюк Т. В. Алгоритмізація та програмування: підручник з грифом МОН України / Т.В. Ковалюк. – Львів : Магнолія-2006, 2013. – 400 с.
24. Шаховська, Н. Б. Алгоритми і структури даних [Текст]: посібник / Н.Б. Шаховська, Р.О. Голощук; - Львів: Магнолія, 2010 – 215с.
25. Веселовська Г.В. та Ходакова, В.Є., 2015. Комп’ютерна графіка. Київ: Кондор.
26. Михайленко В. Є. Інженерна та комп’ютерна графіка. К. : Вища школа, 2002. 332 с.
27. 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.
28. Березовський В.С., Потієнко, В.О. та Завадський, І.О., 2009. Основи комп’ютерної графіки. Київ: BHV.
29. Веселовська Г.В. та Ходакова, В.Є., 2015. Комп’ютерна графіка. Київ: Кондор.
30. Михайленко В. Є. Інженерна та комп’ютерна графіка. К. : Вища школа, 2002. 332 с.

Додаток А

Публікація за темою випускної 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

Conclusion

The developed method of automated generation of rational schemes for cutting rolled materials into parts of leather goods was implemented into a software product. This software product has a friendly interface and does not require additional knowledge of computer science when working with it.

МАМОНОВ Т.А, ЧУПРИНКА В.І.

АЛГОРИТМ ПОБУДОВИ ОПУКЛОЇ ОБОЛОНКИ ДЛЯ МНОГОКУТНИКА

MAMONOV T.A., CHUPRYNKA V.I.

ALGORITHM FOR CONSTRUCTING A CONVEX SHELL FOR A POLYGON

The paper considers an algorithm for solving one of the problems of constructing dense stacks of flat objects close to convex ones, namely, constructing a convex shell for a polygon.
Key words: dense packing, convex shell, polygon.

Вступ

Часта зміна моделей виробів легкої промисловості 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.

Ціллю автоматизації проектування є, зниження матеріальних затрат, скорочення термінів проектування та ліквідація тенденції до збільшення кількості інженерно-технічних робітників, які зайняті проектуванням, підвищення продуктивності їх праці. Це дозволить, по-перше, підвищити якість і скоротити терміни рішення проектних задач за рахунок можливості розглядати як весь об'єкт у цілому, так і взаємозв'язку його елементів. По-друге, розробка структурно-графічних моделей технологічних об'єктів є формалізованим їхнім описом, що дозволяє здійснити перехід до математичних моделей — як основи алгоритмізації інтелектуальних процесів у технологічному проектуванні.

Основна частина

В роботі розглядається одна із задач математичного та програмного забезпечення для побудови щільних укладок плоских об'єктів, близьких до опуклих, а саме побудова опуклої оболонки для многокутника.

Алгоритм побудови опуклої оболонки для многокутника

Постановка задачі. . Нехай маємо многокутника $\boldsymbol{P}$ з координатами вершин (Xp_i, Yp_i), $i=0,1,2..n-1$. Знайти опуклу оболонку $\boldsymbol{R}$ із координатами вершин (Xob_j,Yob_j), $j=0,1,2..p-1$ для многокутника $\boldsymbol{P}$.

Алгоритм побудови опуклої оболонки буде базуватись на властивості векторного добутку двох векторів. $a=(xa,ya)$ та $b=(xb,yb)$, а саме: модуль векторного добутку буде додатнім, якщо найкоротший шлях від вектора a до вектора b буде проти годинникової стрілки.

На рис.1. представлені етапи побудови опуклої оболонки для многокутника та опорної функції. Спочатку вибираємо за початкову вершину, в якій координата X досягає максимуму. Це буде точка P_0 (рис.1.а).

Потім будуємо вектор $a=P_0P_1$ $b=P_0P_2$. Якщо модуль векторного добутку двох векторів. a та b буде менше або дорівнювати нуля при обході контуру проти годинникової стрілки, то вершина P_1 не знаходиться на опуклій оболонці і буде виключена із розгляду (рис. 6.12.б). Аналогічно продовжуємо розгляд контуру многокутника та вилучення з розгляду тих вершин многокутника, які не лежать на опуклій оболонці (рис. 6.21.в) поки не залишиться жодної точки на новому контурі, яка не належить опуклій оболонці (рис. 1.в-г). Побудована опукла оболонка представлена на рис.1.г.

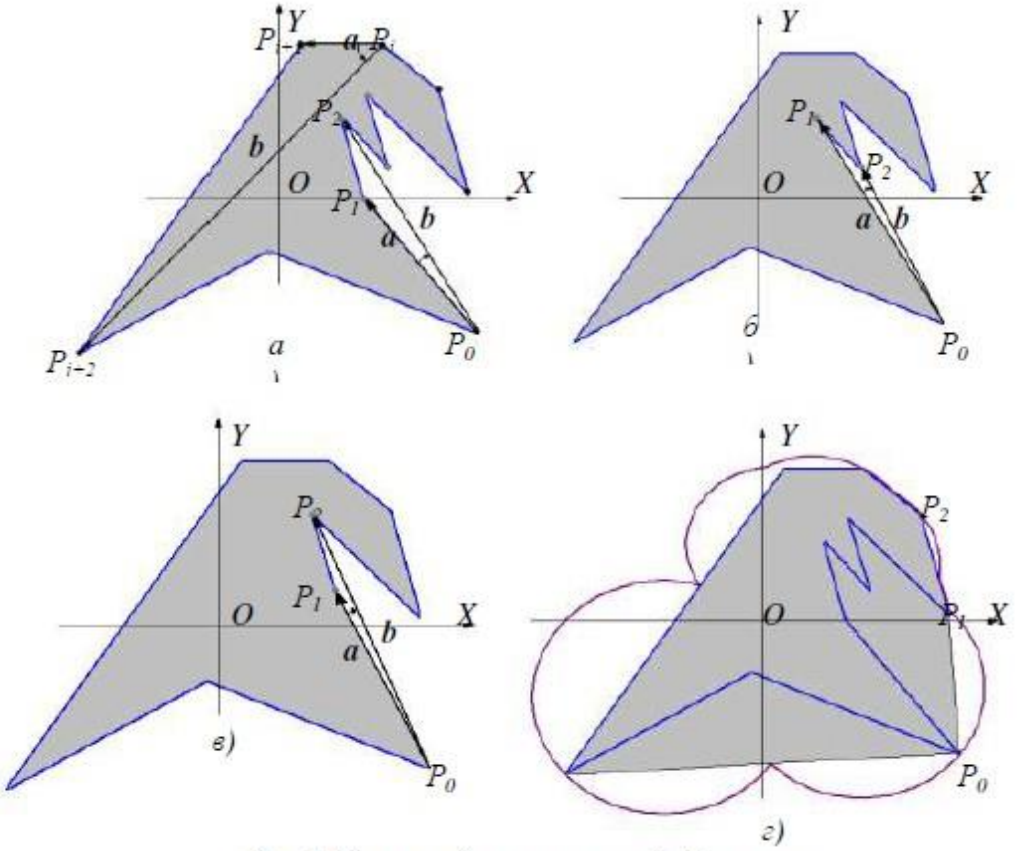

Рис.1. Етапи побудови опуклої оболонки

Алгоритм побудови опуклої оболонки ***R*** для многокутника ***P*** матиме наступні пункти:

1. Визначення на многокутнику такої вершини, що вона завжди входить до вершин на зовнішньому контурі опуклої оболонки. За таку вершину візьмемо будь-яку з вершин многокутника, в якій координата *X* або *Y* досягає екстремуму (максимуму або мінімуму). Нехай за таку вершину приймемо *k* -ту вершину многокутника, для якої $Xp_k = \max_{i=0,1..n-1}(Xp_i)$;

2. В многокутнику ***P*** циклічно переставимо вершини таким чином, що першою вершиною стане *k*-та вершина многокутника, для якої $Xp_k = \max_{i=0,1..n-1}(Xp_i)$;

3. Визначаємо напрям обходу многокутника (порядок, у якому записані координати вершин многокутника при обході многокутника, за годинниковою стрілкою чи проти годинникової стрілки). Для цього знаходимо порядкові номери вершин *k1, k2, k3,* для яких виконуються наступні вирази

$$Yp_{k1} = \max_{i=0,1..n-1}(Yp_i); \quad Xp_{k2} = \min_{i=0,1..n-1}(Xp_i); \quad Yp_{k3} = \min_{i=0,1..n-1}(Yp_i).$$

Якщо виконується наступна нерівність $k1 \le k2 \le k3$, то напрям обходу контуру многокутника проти годинникової стрілки, інакше напрям обходу контуру многокутника за годинниковою стрілкою.

5. Якщо напрям обходу контуру многокутника проти годинникової стрілки, то *Flag1=True,* інакше - *Flag1=False.*

6. Присвоюємо $i=0,\ j=0,\ t=1,\ m=2, Flag2=True,\ Xob_0=Xp_0, Yob_0=Yp_0$;

7. Визначимо $\Delta = (Yp_{i+m} - Yp_i)(Xp_{i+t} - Xp_i) - (Xp_{i+m} - Xp_i)(Yp_{i+t} - Yp_i)$.

Якщо $\Delta<0$ та *Flag1=True* або $\Delta>0$ та *Flag1=False,*

то виконуємо наступні дії

Початок *Flag1=False; q=i+m; m=m+1; l=l+1* ***Кінець***

інакше виконуємо наступні дії

Початок $j=j+1;\ q=i+m+1;\ m=m+1;\ Xob_j=Xp_{i+l}, Yob_j=Yp_{i+l};$

$i=i+l;\ m=2;\ l=1$ ***Кінець***

8. Якщо $q<n-1$, то повертаємося до пункту 7;

9. Присвоюємо $Xob_{j+1}=Xp_0$, $Yob_{j+1}=Yp_0$, $p=j+2$;

10. Якщо *Flag2=False,* то

Початок для $j=0$ до $p-1$ роби

Початок $Xp_j= Xob_j,\ Xp_j= Xob_j$ ***Кінець***

$m=p;$ Повертаємося до пункту 6 , ***Кінець***

інакше ***Кінець.***

Висновки

Запропонований алгоритм побудови опуклої оболонки для многокутника дозволив звести задачу розробки математичного та програмного забезпечення для побудови щільних укладок плоских об'єктів, близьких до опуклих до задачі розробки математичного та програмного забезпечення для побудови щільних укладок для опуклих многокутників. Це значно спростило програмну реалізацію задачі побудови щільних укладок плоских об'єктів, близьких до опуклих

МІНЕНКО М.С., ЧУПРИНКА В.І.

РОЗРОБКА МАТЕМАТИЧНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ІНТЕРАКТИВНОЇ ПОБУДОВИ ЩІЛЬНИХ УКЛАДОК ПЛОСКИХ ОБ'ЄКТІВ

MINENKO M.S., CHUPRYNKA V.I.

DEVELOPMENT OF MATHEMATICAL AND SOFTWARE FOR INTERACTIVE CONSTRUCTION OF DENSE LAYERS OF FLAT OBJECTS

The article is devoted to the development of mathematical and software for interactive construction of dense stacks. The software has a user-friendly interface and does not require additional computer science knowledge when working with it.
Key words: dense stacking, interactive construction, angle method

Вступ

Часта зміна моделей суконь для дівчаток 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.

Ціллю автоматизації проектування є, зниження матеріальних затрат, скорочення термінів проектування та ліквідація тенденції до збільшення кількості інженерно-технічних робітників, які зайняті проектуванням, підвищення продуктивності їх праці. Це дозволить, по-перше, підвищити якість і скоротити терміни рішення проектних задач за рахунок можливості розглядати як весь об'єкт у цілому, так і взаємозв'язку його елементів. По-друге, розробка структурно-графічних моделей технологічних об'єктів є формалізованим їхнім описом, що дозволяє здійснити перехід до математичних моделей — як основи алгоритмізації інтелектуальних процесів у технологічному проектуванні.

Основна частина

Загальну технологічну постановку задачі проектування найщільнішої укладки можна сформулювати наступним чином: розмістити деталі в

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 ПОБУДОВИ ЩІЛЬНИХ УКЛАДОК ПЛОСКИХ ОБ'ЄКТІВ

В.І. Чупринка, доктор технічних наук, професор
Київський національний університет технологій та дизайну
Т.А. Мамонов, магістрант
Київський національний університет технологій та дизайну
М.С. Міненко, магістрант
Київський національний університет технологій та дизайну

Ключові слова: математичне та програмне забезпечення, щільні укладки, легка промисловість.

Застосування якісного програмного забезпечення для автоматизованого проектування раціональних схем розкрою матеріалів на деталі виробів легкої промисловості підвищить процент ефективного використання матеріалу при розкрою та зменшить кількість відходів, які потрібно буде утилізувати.

Постановка задачі. Припустимо, що многокутник S решітчасто розміщується на площині. Розглядається подвійні решітчасті укладки многокутників із заданою орієнтацією $S(0)$ і $S(\pi)$ повернутих на кут π і розв'язується задача пошуку найщільнішої серед них.

Зформулюємо вказану задачу більш чітко:

Задача *Серед подвійних решітчастих укладок W многокутників $S(0)$ і $S(\pi)$ знайти таку $W^* = W(\vec{a}_1^{\,*}, \vec{a}_2^{\,*}, \vec{g}^{\,*})$, щоб щільність $\delta_S(W^*)$ подвійної укладки многокутників $S(0)$ і $S(\pi)$, виконаної за цією решіткою задовільнила співвідношенню*

$$\delta_S(W^*) = \max_W \delta_S(W). \qquad (1)$$

Враховуючи, що $|S|$ – площа многокутника є сталою, задачу можна зформулювати по іншому:

Серед подвійних решіток $W = W(\vec{a}_1, \vec{a}_2, \vec{g})$, допустимих для укладки фігур $S(0)$ і $S(\pi)$, знайти таку $W^* = W(\vec{a}_1^{\,*}, \vec{a}_2^{\,*}, \vec{g}^{\,*})$, детермінант якої має мінімальне значення.

$$detW^* = detW(\vec{a}_1^{\,*}, \vec{a}_2^{\,*}, \vec{g}^{\,*}) = \min detW(\vec{a}_1, \vec{a}_2, \vec{g}), \qquad W \in K^{(3)} \quad (2)$$

Задача (2) являється задачею математичного програмування, для розв'язання якої необхідно побудувати і описати множину допустимих рішень $K^{(3)}$ і вказати метод пошуку W^* на цій множині.

Розв'язання цих питань може бути одержано на основі вектор-функцій щільного розміщення фігур і їх годографів.

Основою для побудови областей допустимих розв'язків у всіх зформульованій задачі являються умови взаємного неперетину фігур в укладці.

Алгоритм пошуку щільних укладок

1. Перевіряємо на опуклість многокутник;
2. Обчислюємо площу многокутника;
3. Розраховуємо координати вершин годографа Γ_{11} вектор-функції щільного розміщення двох одинакових і одинаково орієнтованих многокутників S;
4. Розраховуємо координати вершин годографа Γ_{12} вектор-функції щільного розміщення двох одинакових многокутників $S(0)$ і $S(\pi)$;
5. Задаємо $i=1$, $t_i=0$, і обчислюємо стартові значення змінних j, k, u, v, а також стартові значення параметрів τ_j, τ_k, τ_u, τ_v.
6. Знаходимо інтервал допустимих значень параметрів t_i, τ_j, τ_k, τ_u, τ_v, в полі функціонування класу K_{ijuv}, де $t_i^0 \le t_i \le t_i^1$, $\tau_j^0 \le \tau_j \le \tau_j^1$, $\tau_k^0 \le \tau_k \le \tau_k^1$, $\tau_u^0 \le \tau_u \le \tau_u^1$, $\tau_v^0 \le \tau_v \le \tau_v^1$.
7. Обчислюємо коефіцієнти цільової функції $detW(\vec{a},\vec{p},\vec{g})$, яка є квадратичною функцією параметра $t_i \in (t_i^0, t_i^1)$, де $t_i^0, t_i^1 \in [0;1]$.
8. Обчислюємо значення параметру t_i^p, яке відповідає критичній точці цільової функції.
9. Обчислюємо значення цільової функції $detW(\vec{a},\vec{p},\vec{g})$ в точках t_i^0, t_i^1, та в точці t_i^p, якщо вона є внутрішньою точкою (t_i^0, t_i^1);
10. Визначаємо екстремум у класі K_{ijuv}, та заносимо у масив локальних екстремумів відповідні значення $i, j, u, t_i, \tau_j, \tau_u$, і $detW(\vec{a},\vec{p},\vec{g})$.
11. Перевіряємо $t_i^1=1, \tau_j^1=1, \tau_k^1=1, \tau_u^1=1, \tau_v^1=1$.
12. Ті з індексів i, j, k, u, v для яких відповідне значення параметру τ досягло одиниці, збільшуємо на одиницю, а параметр τ задаємо рівним нулю;
13. Якщо $i \le n$, то переходимо на крок 6, інакше на крок 14; n – кількість вершин многокутника S.
14. В масиві локальних оптимумів, шляхом порівняння значень $detW(\vec{a},\vec{p},\vec{g})$, знаходимо глобальний оптимум і строку параметрів та відповідне значення цільової функції видаємо як розв'язок задачі.

Висновки

В роботі запропонований методи та алгоритми для проєктування щільних укладок плоских геометричних об'єктів зі складною конфігурацією зовнішнього контуру, які реалізовані в програмний продукт.

Додаток Б

Листинг програмного продукту

```
unit UnMamonov;

interface

uses
  Windows, Messages, SysUtils,  Classes, Graphics, Controls, Forms,
  Dialogs, Menus, ExtCtrls,Printers,StdCtrls, ComCtrls;

type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Open1: TMenuItem;
    N1: TMenuItem;
    Exit1: TMenuItem;
    OpenDialog1: TOpenDialog;
    ScrollBox1: TScrollBox;
    Image2: TImage;
    Button1: TButton;
    Edit1: TEdit;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    Image3: TImage;
    TabSheet3: TTabSheet;
    Image4: TImage;
    Image1: TImage;
    TabSheet4: TTabSheet;
    Image5: TImage;
    GroupBox1: TGroupBox;
    Button3: TButton;
    Button4: TButton;
    RadioGroup1: TRadioGroup;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
    TabSheet5: TTabSheet;
    ScrollBox2: TScrollBox;
    Button2: TButton;
```

```
    Image6: TImage;
    procedure Open1Click(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure z(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);



  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}
   type mas=array[0..100] of real;
        mas1=array[1..40] of mas;
        mas2=array[0..100] of int64;
        mas3=array[1..40] of mas2;
        var x,y:mas1;
     //      dl_det,sh_det:mas;
          model:string[40];
          numdet,koldet,imin,Nob,q,ng:integer;
          namedet:array[1..40] of string[20];
          nn,kolpointdet:array[1..40] of integer;
          xm,ym:mas3;
          Xob,Yob,XobR,YobR:mas2;
          xc,yc,dl,sh,Most,XaR,YaR,XbR,YbR,XaF,YaF,XbF,YbF:int64;
          xcdet,ycdet,dl_det,sh_det: array[1..40] of int64;
         // Sh,Dl,Most:int64;
          i:word;
          mmxy,mxy,mx,dl_dt,sh_dt,Pr_Isp:real;
```

```
  PrBTN2,prBTN3,PrBTN4,PrBTN5,Prmouse,Pr_L,Pr_R:Boolean ;
xq,yq: integer;


  Procedure Skv(n:integer; Var x,y:array of Int64; Var s:Int64);
   Var i:integer;
    Begin
     s:=0;
     For i:=0 to n-2 do
     s:=s+x[i]*y[i+1]-x[i+1]*y[i];
     s:=Round(abs(s)/2)
    End;


Procedure SCALE(xmax,xmin,ymax,ymin:int64;sh,dl:word; var mxy:real);
var Rsh,Rdl:int64;mx:real;
begin
Rsh:=xmax-xmin;
Rdl:=ymax-ymin;
mx:=(sh-10)/Rsh;
mxy:=(dl-10)/Rdl;
If mx<mxy then mxy:=mx;
end;

Procedure SCALE_image2(var mxy:real);
var Rsh,Rdl:real;mx:real;
begin
Rsh:=sh_dt;
Rdl:=dl_dt;
mx:=40/Rsh;
mxy:=40/Rdl;
If mx<mxy then mxy:=mx;
end;


procedure Graph(col:integer;mmxy:real; n:word; mxy:real; xm,ym:array of int64;
k:byte; xc,yc:integer);
var Xr,Yr:array[0..100] of integer; i:word;
Begin
for i:=0 to n-1 do {Перераховуємо реальні координати в екоанні} begin
```

```
      xr[i]:=round(xm[i]*mmxy+xc);
      yr[i]:=round(ym[i]*mmxy+yc);
      end;
      with form1.image1.canvas do  begin
      case col of
      1:  pen.Color:=ClRed; {Встановлюємо колір пера}
      2:  pen.Color:=ClBlack;
      3:  pen.Color:=ClGray;
      4:  pen.Color:=ClYellow;
      5:  pen.Color:=ClPurple;
      6:  pen.Color:=ClSilver;
      7:  pen.Color:=ClGreen;
      8:  pen.Color:=ClAqua;
      9:  pen.Color:=ClBlue;
      10:  pen.Color:=ClFuchsia;
      11:  pen.Color:=ClTeal;
      12:  pen.Color:=ClNavy;
      13:  pen.Color:=ClMaroon;
      14:  pen.Color:=ClOlive
      else pen.Color:=TColor(RGB(255,128,64));
      end;
for i:=0 to n-1 do
    if i=0 then moveto(xr[i],yr[i]) else lineto(xr[i],yr[i]); {Рисуемо контури деталі}
end;
End;



Procedure GrdM(n:integer; Var xr,yr:array of Int64;
  dxc,dyc,color:integer; dx,dy:Int64;pr:integer);
   Var i:integer;
    xp,yp:array[0..200] of integer;
    Begin
      for i:=0 to n-1 do
       begin
        xp[i]:=round((xr[i]*pr+dx)*mxy)+dxc;
       yp[i]:=round((dy+yr[i]*pr)*mxy)+dyc;
       end;
      With Form1.Image1,Canvas Do
       begin
          If pr=1 then Pen.Color:=ClFuchsia{ClGreen}
         else Pen.Color:=ClLime;
```

```
        Pen.Mode:=pmXOR;
      MoveTo(xp[0],yp[0]);
      for i:=1 to n-1 do
      LineTo(xp[i],yp[i]);
      end;
    End;


procedure maxmin(var kd: integer;kpd:integer;x:array of int64;var
xi:int64;p:integer);
var i:word;
 begin
for i:=0 to kpd-1 do begin
if i=0 then begin
xi:=x[i];
kd:=i;
end
else begin
if x[i]*p>xi*p then begin
xi:=x[i];
kd:=i;
end;
end;
end;
end;

  procedure XcYc(n:integer;x,y: array of int64; var xc,yc:int64);
var nxi:integer; xi,xe,yi,ye:int64;
begin
    maxmin(nxi,n,x,xi,-1);
    maxmin(nxi,n,y,yi,-1);
    maxmin(nxi,n,x,xe,1);
    maxmin(nxi,n,y,ye,1);
    xc:=round((xi+xe)/2);
    yc:=round((yi+Ye)/2);
end;

 Procedure Obxod(n:integer; Var x,y:array of Int64);
     Var k,i,nxi,nxe,nyi,nye:integer;
        xr,yr:mas2;
        xi,yi,xe,ye:Int64;
```

```
    pr:boolean;
  Begin
   k:=1;
  MaxMin(nye,n,y,ye,k);
  MaxMin(nxe,n,x,xe,k);
   k:=-1;
  MaxMin(nyi,n,y,yi,k);
  MaxMin(nxi,n,x,xi,k);
  pr:=(nxi<=nyi)and(nyi<=nxe)and(nxe<=nye);
  if  not pr then
   for i:=n-1 downto 0 do
    begin
     xr[n-i-1]:=x[i];
     yr[n-i-1]:=y[i];
    end;
    for i:=0 to n-1  do
    begin
     x[i]:=xr[i];
     y[i]:=yr[i];
    end;
   End;


procedure TForm1.Open1Click(Sender: TObject);

procedure vvod;
var f:system.text;
filename:string;
xi,yi,xe,ye,xmax,ymax,xmin,ymin:int64;
i,j:word;
nxi:integer;
xrab,yrab:real;
A:string;
begin
if opendialog1.execute then
begin
filename:=opendialog1.filename;
system.Assign(f,filename);
reset(f);
readln(f,model);
caption:='модель'+model;
readln(f,a);
```

```
readln(f,koldet);
for i:=1 to koldet do
readln(f,namedet[i]);
for i:=1 to koldet do
begin
  readln(f,kolpointdet[i]);
  nn[i]:=kolpointdet[i];
  end;
for i:=1 to koldet do
 begin
  for j:=0 to nn[i]-1 do
  begin
  readln(f,xrab,yrab);
  if kolpointdet[i]<>nn[i] then continue;
  if (xrab=999)and (yrab=999) then
  begin
    kolpointdet[i]:=j;
    continue
    end;
    xm[i,j]:=round(xrab*100);
    ym[i,j]:=round(yrab*100);

        end;
         nn[i]:=kolpointdet[i];
            end;
        for i:=1 to koldet do begin

    maxmin(nxi,kolpointdet[i],xm[i],xi,-1);
    maxmin(nxi,kolpointdet[i],ym[i],yi,-1);
    maxmin(nxi,kolpointdet[i],xm[i],xe,1);
    maxmin(nxi,kolpointdet[i],ym[i],ye,1);
    dl_det[i]:=xe-xi;
     sh_det[i]:=ye-yi;
     if  i=1  then  begin
     dl_dt:=dl_det[i];
     sh_dt:=sh_det[i];
     end
      else
      begin
         if dl_dt<dl_det[i] then dl_dt:=dl_det[i];
         if sh_dt<sh_det[i] then sh_dt:=sh_det[i];
      end;
```

```
      if i=1 then
       begin
       xmax:=xe;
       ymax:=ye;
       xmin:=xi;
       ymin:=yi;
       xcdet[1]:=round((xe+xi)/2);
       ycdet[1]:=round((ye+yi)/2);

       end
       else
       begin
       xcdet[I]:=round((xe+xi)/2);
       ycdet[I]:=round((ye+yi)/2);
      //  DlDet[i]:=Xe-Xi;
      //  ShDet[i]:=Ye-Ye;
       if xmax<xe then xmax:=xe;
       if ymax<ye then ymax:=ye;
       if xmin>xi then xmin:=xi;
       if ymin>yi then ymin:=yi;
       end
         end;  end;
         xc:=round((xmax+xmin)/2);
         yc:=round((ymax+ymin)/2);
         dl:=xmax-xmin;
         sh:=ymax-ymin;
      for i:=1 to koldet do
      begin
       for j:=0 to kolpointdet[i]-1 do
        begin
        xm[i,j]:=xm[i,j]-xcdet[i];
        ym[i,j]:=ym[i,j]-ycdet[i];
        end;
{        xcdet[i]:=xcdet[i]+xc;
        ycdet[i]:=ycdet[i]+yc;}
        SCALE(xmax,xmin,ymax,ymin,image1.Width,image1.height, mxy);

        Obxod(KolPointDet[i],xm[i],ym[i]);
       end ;
        system.Close(f);
      end;
```

```
   begin
   TabSheet1.Visible:=True;
   TabSheet2.Visible:=True;
   TabSheet3.Visible:=True;
   TabSheet4.Visible:=True;
   TabSheet1.TabVisible:=True;
   TabSheet2.TabVisible:=True;
   TabSheet3.TabVisible:=True;
   TabSheet4.TabVisible:=True;
   vvod;
   ScrollBox1.Visible:=true;
     With form1.Image2 .Canvas  do
begin
 pen.Color :=clred;
 form1.Image2 .Height :=koldet*55+10;
 scale_image2(mmxy);
 for i:=1 to koldet do  begin
 XcYc(kolpointdet[i],xm[i],ym[i],xc,yc);

Rectangle(5,5+55*(i-1),55,50+55*(i-1))  ;
Graph_im2(i,mmxy,kolpointdet[i],  xm[i],ym[i], 27,round(27+55*(i-1)),xc,yc);
end;
end;

end;

procedure scale_im2(var mx:real);
var   my:real ;
begin
mx:=form1.Image2.Width/dl_dt ;
my:=form1.Image2.Height /sh_dt ;
if mx>my then mx:=my;

end;


 Procedure VipDet(n:integer; x,y:array of Int64; Var Xob,Yob:array of Int64;
              Var Nob:integer);
    Var
     i,l,m,p,q:integer;
       delta:real;
```

```
        pr:boolean;
      Begin
       p:=0;
       Xob[p]:=X[0];
       Yob[p]:=Y[0];
       Nob:=n;
      Repeat
      i:=0;
       m:=2; l:=1;
       pr:=true;

       Repeat
        delta:=(y[i+m]-y[i])*(x[i+L]-x[i])-
              (y[i+L]-y[i])*(x[i+m]-x[i]);
        if delta<=0 then
          begin
           pr:=false;
           q:=i+m;
           m:=m+1;
           L:=L+1;
//           q:=i+m;
          end
        else
         begin
         p:=p+1;
          q:=i+m+1;
          Xob[p]:=X[i+L];
          Yob[p]:=Y[i+L];
          i:=i+L;
//          q:=i+m;
          m:=2;
          L:=1

         end;
        Until q>Nob-1;
        Xob[p+1]:=X[0];
        Yob[p+1]:=Y[0];
        Nob:=p+2;
        p:=0;
        For i:=0 to Nob-1 do
         begin
          X[i]:=Xob[i];
```

```
      Y[i]:=Yob[i];
    end;
  Until pr;

End;

 Procedure ForDet(n,ni:integer; Var x,y,xn,yn:array of Int64);
          Var i,k:integer;
          begin
            k:=n-ni-1;
            for i:=0 to k do
              begin
              xn[i]:=x[ni+i];
               yn[i]:=y[ni+i]
              end;
             for i:=0 to ni do
              begin
              xn[k+i]:=x[i];
               yn[k+i]:=y[i]
              end;
          end;

        Procedure GodVip(nn,nv:integer; Var xn,yn,xv,yv,
                  xg,yg:array of Int64; Var k:integer);
          Var i,j:integer;
            s:real;
          begin
      {    writeln(nn:4,nv:4,xn[1]:5,yn[1]:5,xv[1]:5,yv[1]:5);}
          i:=0;
          j:=0;
          k:=-1;
          xn[nn]:=xn[1];
          yn[nn]:=yn[1];
          xv[nv]:=xv[1];
          yv[nv]:=yv[1];
          xn[nn+1]:=xn[2];
          yn[nn+1]:=yn[2];
          xv[nv+1]:=xv[2];
          yv[nv+1]:=yv[2];

          repeat
           k:=k+1;
```

```
                xg[k]:=xv[i]-xn[j];
                yg[k]:=yv[i]-yn[j];
                s:=(xn[j]-xn[j+1])*(yv[i+1]-yv[i])-
                  (yn[j]-yn[j+1])*(xv[i+1]-xv[i]);
                    if s<=0 then i:=i+1
                    else j:=j+1;
{                 writeln(k:3,xg[k]:7:2,yg[k]:7:2);
                readln;                        }
               until ((k>nn-1) and (xg[0]=xg[k]) and
                    (k>nv-1) and (yg[0]=yg[k]));
                    k:=k+1
           end;


procedure TForm1.Exit1Click(Sender: TObject);
begin
close;
end;



procedure TForm1.FormCreate(Sender: TObject);
var  i:Integer ;
begin
PageControl1.ActivePage:=TabSheet5;
Prmouse:=False;
PrBTN2 :=False ;
PrBTN3 :=False ;
PrBTN4 :=False ;
// Image2.Height:=2000;
form1.Height :=600;
form1.Width :=800;
form1.Left :=0;
form1.Top :=0;
 q:=-1;


end;

procedure TForm1.z(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
```

```
  var  nxi,ni,ne,i,j:integer; xc,yc,xi,xe,yi,ye:int64;
  xr,yr,xrp,yrp,xn,yn,xg,yg:mas2;

       function SETnumberDet(xcur,ycur:integer):integer;
       var i:integer;

       begin
        Edit1.Visible:=true;
             //Image2.Height:=2000;
        for i:=1 to koldet do
         if (xcur>5) and (xcur<55) and (ycur>5*i+50*(i-1)) and (ycur<55*i) then
         begin
          SETnumberDet:=i;
          break;
         end;
       end;
Begin
PageControl1.Visible:=True;
GroupBox1.Visible:=True;
RadioGroup1.Visible:=True;
RadioButton1.Visible:=True;
RadioButton2.Visible:=True;
{If button=MbLeft then
 with image2 do
  begin
   numDET:=SETnumberDet(x,y);
   Edit1.Text:=namedet[numDET];
   SCALE(round(dl_det[numDet]/2),round(-
dl_det[numDet]/2),round(sh_det[numDet]/2),round(-
sh_det[numDet]/2),image1.Height-50,image1.Width-50, mxy);
   with image1.Canvas do
    begin
    pen.Color:=ClWhite;
    brush.Color:=ClWhite;
    brush.Style:=BsSolid;
     rectangle(0,0,image1.Width,image1.Height);
     i:=numdet ;
     XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);

Graph_im1(numdet,mxy,kolpointdet[numdet],xm[numdet],ym[numdet],round(image1
.Width/2),round(image1.Height/2), xc,yc);
    end;
```

```
 end
else
begin }
  with image2 do
   begin
    numDET:=SETnumberDet(x,y);
   end;
   Edit1.Text:=namedet[numDET];
    SCALE(round(dl_det[numDet]/2),round(-
dl_det[numDet]/2),round(sh_det[numDet]/2),round(-
sh_det[numDet]/2),image1.Height-50,image1.Width-50, mxy);
     with image1.Canvas do
      begin
      pen.Color:=ClWhite;
      brush.Color:=ClWhite;
      brush.Style:=BsSolid;
       rectangle(0,0,image1.Width,image1.Height);
       i:=numdet ;
       XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);
       maxmin(nxi,kolpointdet[numdet],xm[numdet],xi,-1);
       ForDet(kolpointdet[numdet],nxi,xm[numdet],ym[numdet],XobR,YobR);
       VipDet(kolpointdet[numdet],XobR,Yobr,Xob,Yob,Nob);
       VipDet(Nob,Xob,Yob,Xob,Yob,Nob);
       XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);
       For i:=0 to kolpointdet[numdet]-1 do
        begin
         xm[numdet,i]:=xm[numdet,i]-xc;
        ym[numdet,i]:=ym[numdet,i]-yc;
        end;
       For i:=0 to Nob-1 do
        begin
        Xob[i]:=Xob[i]-Xc;
         Yob[i]:=Yob[i]-Yc;
        end;
       XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);

Graph_im1(numdet,mxy,kolpointdet[numdet],xm[numdet],ym[numdet],round(image1
.Width/2),round(image1.Height/2), xc,yc);

Graph_im1(2,mxy,Nob,Xob,Yob,round(image1.Width/2),round(image1.Height/2),
xc,yc);
    end;
```

```
  with image3.Canvas do
   begin
   pen.Color:=ClWhite;
   brush.Color:=ClWhite;
   brush.Style:=BsSolid;
    rectangle(0,0,image3.Width,image3.Height);
    i:=numdet ;
 //   XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);

Graph_im3(numdet,mxy/4,kolpointdet[numdet],xm[numdet],ym[numdet],round(imag
e1.Width/2),round(image1.Height/2), xc,yc);


      MaxMin(ni,Nob,Xob,xi,-1);
      ForDet(Nob,ni,Xob,Yob,xr,yr);
      MaxMin(ne,Nob,xr,xe,1);
      ForDet(Nob,ne,xr,yr,xn,yn);
      GodVip(Nob,Nob,xr,yr,xn,yn,xg,yg,ng);
   //   XcYc(ng,xg,yg,xc,yc);
      Graph_im3(2,mxy/4,ng,xg,yg,round(image1.Width/2),round(image1.Height/2),
xc,yc);
     for i:=0 to ng-1 do
   begin
   for j:=0 to kolpointdet[numdet]-1 do
    begin
     xr[j]:=xm[numdet,j]+xg[i];
     yr[j]:=ym[numdet,j]+yg[i];
    end;
  //  XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);

Graph_im3(1,mxy/4,kolpointdet[numdet],xr,yr,round(image1.Width/2),round(image1.
Height/2), xc,yc);

   end;
 end;

  with image4.Canvas do
   begin
   pen.Color:=ClWhite;
   brush.Color:=ClWhite;
   brush.Style:=BsSolid;
```

```
    rectangle(0,0,image4.Width,image4.Height);
   { pen.Color:=ClRed;
    ellipse(40,40,80,80);}
    i:=numdet ;
   // XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);

Graph_im4(numdet,mxy/4,kolpointdet[numdet],xm[numdet],ym[numdet],round(imag
e1.Width/2),round(image1.Height/2), xc,yc);
      MaxMin(ni,Nob,Xob,xi,-1);
      ForDet(Nob,ni,Xob,Yob,xr,yr);
      For i:=0 to Nob-1 do
       begin
        xrp[i]:=-xr[i];
        yrp[i]:=-yr[i];
       end;
      MaxMin(ne,Nob,xrp,xe,1);
      ForDet(Nob,ne,xrp,yrp,xn,yn);
      GodVip(Nob,Nob,xn,yn,xr,yr,xg,yg,ng);
  //   XcYc(ng,xg,yg,xc,yc);
      Graph_im4(2,mxy/4,ng,xg,yg,round(image1.Width/2),round(image1.Height/2),
xc,yc);
  for i:=0 to ng-1 do
    begin
    for j:=0 to kolpointdet[numdet]-1 do
     begin
      xr[j]:=-xm[numdet,j]+xg[i];
      yr[j]:=-ym[numdet,j]+yg[i];
     end;
  //   XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);

Graph_im4(1,mxy/4,kolpointdet[numdet],xr,yr,round(image1.Width/2),round(image1.
Height/2), xc,yc);
   end;
  end;
  with image5.Canvas do
    begin
    pen.Color:=ClWhite;
    brush.Color:=ClWhite;
    brush.Style:=BsSolid;
     rectangle(0,0,image5.Width,image5.Height);
    end
```

```
 end;



procedure TForm1.Button1Click(Sender: TObject);
begin
close ;
end;
(*
{Малює еліпс в вершинах прямокутника}
procedure ellips(r,n:integer;x,y:array of int64; xc,yc,xcIm,ycIm:int64;mxy:real);
      var i:integer;  xe,ye:int64;
      begin
      with form1.image1.Canvas do
      begin
      pen.color:=ClRed;
      for i:=0 to n-1 do
      begin
      xe:=round((x[i]-xc)*mxy+xcIm);
      ye:=round((y[i]-yc)*mxy+ycIm);
      ellipse(xe-r,ye-r, xe+r,ye+r);
      end;
      end;
end;

procedure ellipsDEL(r,n1,n:integer;x,y:array of int64;
xc,yc,xcIm,ycIm:int64;mxy:real);
      var i:integer;  xe,ye:int64;
      begin
      with form1.image1.Canvas do
      begin
      pen.color:=ClWhite;
      xe:=round((x[n1]-xc)*mxy+xcIm);
      ye:=round((y[n1]-yc)*mxy+ycIm);
      ellipse(xe-r,ye-r, xe+r,ye+r);
      end;

end;*)

procedure ellipsONE(r:integer;x,y:int64; xc,yc,xcIm,ycIm:int64;mxy:real);
      var i:integer;  xe,ye:int64;
      begin
```

```
        with form1.image1.Canvas do
        begin
        pen.color:=ClRed;
        xe:=round((x-xc)*mxy+xcIm);
        ye:=round((y-yc)*mxy+ycIm);
        ellipse(xe-r,ye-r, xe+r,ye+r);
        end;

end;


procedure ellipsONE5(r:integer;x,y:int64; xc,yc,xcIm,ycIm:int64;mxy:real);
        var i:integer;  xe,ye:int64;
        begin
        with form1.image5.Canvas do
        begin
        pen.color:=ClRed;
        xe:=round((x-xc)*mxy+xcIm);
        ye:=round((y-yc)*mxy+ycIm);
        ellipse(xe-r,ye-r, xe+r,ye+r);
        end;

end;


Function PointCrossTwoLine(Xa,Ya,Xb,Yb,Xc,Yc,Xd,Yd:Int64; var
x0,y0:Int64):boolean;
   Var  A1,B1,C1,A2,B2,C2,Ra,Rb,Rc,Rd,Rab,Rcd,D1,D2,D0:Int64;
        i:integer;
        B:boolean;

  Begin
     PointCrossTwoLine:=False;
     A1:=ya-yb;
     B1:=xb-xa;
     C1:=yb*(xa-xb)-xb*(ya-yb);
     A2:=yc-yd;
     B2:=xd-xc;
     C2:=yd*(xc-xd)-xd*(yc-yd);
     Ra:=A2*Xa+B2*Ya+C2;
     Rb:=A2*Xb+B2*Yb+C2;
     Rc:=A1*Xc+B1*Yc+C1;
```

```
    Rd:=A1*Xd+B1*Yd+C1;
    Rab:=Ra*Rb;
    Rcd:=Rc*Rd;
    B:=((Rcd<0)and(Rab<0))or((Rcd<0)and(Rab=0))or((Rcd=0)and(Rab<0));
    If B then
     begin
      PointCrossTwoLine:=True;
      D0:=A1*B2-A2*B1;
      D1:=C2*B1-C1*b2;
      D2:=C1*A2-C2*A1;
      X0:=Round(D1/D0);
      Y0:=Round(D2/D0);
     end;
  End;



procedure TForm1.Button3Click(Sender: TObject);
var  nxi,ni,ne,i,j,p,ng,ngp,npg:integer; xc,yc,xc1,yc1,xi,xe,yi,ye,x0,y0:int64;
  xr,yr,xn,yn,xg,yg,xpg,ypg,xgp,ygp,xp,yp,xrp,yrp,xrr,yrr:mas2;
  Spar,SDet,Ykl:real;
begin
 p:=0;
 q:=q+1;
 If q>=ng-1 then q:=0;
 with image5.Canvas do
 begin
 pen.Color:=ClWhite;
 brush.Color:=ClWhite;
 brush.Style:=BsSolid;
   rectangle(0,0,image5.Width,image5.Height);
   XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);

Graph_im5(numdet,mxy/4,kolpointdet[numdet],xm[numdet],ym[numdet],round(imag
e5.Width/2),round(image5.Height/2), xc,yc);
   maxmin(nxi,kolpointdet[numdet],xm[numdet],xi,-1);
   ForDet(kolpointdet[numdet],nxi,xm[numdet],ym[numdet],XobR,YobR);
   VipDet(kolpointdet[numdet],XobR,Yobr,Xob,Yob,Nob);
   VipDet(Nob,Xob,Yob,Xob,Yob,Nob);
   MaxMin(ni,Nob,Xob,xi,-1);
   ForDet(Nob,ni,Xob,Yob,xr,yr);
   MaxMin(ne,Nob,xr,xe,1);
```

```
  ForDet(Nob,ne,xr,yr,xn,yn);
  GodVip(Nob,Nob,xr,yr,xn,yn,xg,yg,ng);
for j:=0 to kolpointdet[numdet]-1 do
     begin
     xrr[j]:=xm[numdet,j]+xg[q];
      yrr[j]:=ym[numdet,j]+yg[q];
     end;
 //
Graph_im5(16,mxy/4,ng,xgp,ygp,round(image5.Width/2),round(image5.Height/2),
xc,yc);
       Graph_im5(numdet,mxy/4,kolpointdet[numdet],xrr,yrr,round(image5.Width
/2),round(image5.Height/2), xc,yc);

  If RadioButton1.Checked then
   begin
//    XcYc(ng,xg,yg,xc1,yc1);
    Graph_im5(1,mxy/4,ng,xg,yg,round(image5.Width/2),round(image5.Height/2),
xc,yc);
    for j:=0 to ng-1 do
     begin
     xgp[j]:=xg[j]+xg[q];
      ygp[j]:=yg[j]+yg[q];
     end;
{    for j:=0 to kolpointdet[numdet]-1 do
     begin
      xr[j]:=xm[numdet,j]+xg[q];
      yr[j]:=ym[numdet,j]+yg[q];
     end;    }
     Graph_im5(2,mxy/4,ng,xgp,ygp,round(image5.Width/2),round(image5.Height/2),
xc,yc);

//      XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);
//      Graph_im5(numdet,mxy/4,kolpointdet[numdet],xr,yr,round(image5.Width
/2),round(image5.Height/2), xc,yc);
     For i:=0 to ng-2 do
      For j:=0 to ng-2 do
       begin
        if
PointCrossTwoLine(Xg[i],Yg[i],Xg[i+1],Yg[i+1],Xgp[j],Ygp[j],Xgp[j+1],Ygp[j+1],
x0,y0) then
         begin
          p:=p+1;
```

```
          xp[p]:=x0;
          yp[p]:=y0;
         end
       end;
    For i:=1 to p do
     begin

ellipsONE5(2,xp[i],yp[i],xc,yc,round(image5.Width/2),round(image5.Height/2),mxy/4
);
     end ;
     ellipsONE5(2,0,0,xc,yc,round(image5.Width/2),round(image5.Height/2),mxy/4);

ellipsONE5(2,xg[q],yg[q],xc,yc,round(image5.Width/2),round(image5.Height/2),mxy/
4);

     Pen.Color:=ClBlue;
     MoveTo(round(Xc*mxy/4+image5.Width/2),round(Yc*mxy/4+image5.Height/2));

LineTo(round((Xp[2]+Xc)*mxy/4+image5.Width/2),round((Yp[2]+Yc)*mxy/4+image
5.Height/2));

LineTo(round((Xg[q]+Xc)*mxy/4+image5.Width/2),round((Yg[q]+Yc)*mxy/4+image
5.Height/2));

LineTo(round((Xp[1]+Xc)*mxy/4+image5.Width/2),round((Yp[1]+Yc)*mxy/4+image
5.Height/2));
     LineTo(round(Xc*mxy/4+image5.Width/2),round(Yc*mxy/4+image5.Height/2));

     Spar:=Abs(Xp[1]*Yp[2]-Xp[2]*Yp[1])/100000000;
    (* Sdet:=0;
     for i:=0 to kolpointdet[numdet]-2  do
      Sdet:=Sdet+xm[numdet,i]*ym[numdet,i+1]-ym[numdet,i]*xm[numdet,i+1];
      Sdet:=abs(Sdet/200000000);
      Ykl:= Sdet/Spar*100;
      Font.Name:='Ariel';
      Font.Size:=10;
{       Font.Style:=[fsBold];,fsItalic];}
      Font.Color:=clBlack;
      TextOut(image5.Width-250,20,'Щільність
укладки='+FloatToStrF(Ykl,ffFixed,6,2)+'%');
      TextOut(image5.Width-250,40,'Площа
деталі='+FloatToStrF(SDet,ffFixed,6,2)+'кв.дм');
```

```
      TextOut(image5.Width-250,60,'Площа
паралелограму='+FloatToStrF(Spar,ffFixed,6,2)+' кв.дм');*)

   for j:=0 to kolpointdet[numdet]-1 do
    begin
     xr[j]:=xm[numdet,j]+xp[1];
     yr[j]:=ym[numdet,j]+yp[1];
    end;

Graph_im5(numdet,mxy/4,kolpointdet[numdet],xr,yr,round(image5.Width/2),round(i
mage5.Height/2), xc,yc);
   for j:=0 to kolpointdet[numdet]-1 do
    begin
     xr[j]:=xm[numdet,j]+xp[2];
     yr[j]:=ym[numdet,j]+yp[2];
    end;

Graph_im5(numdet,mxy/4,kolpointdet[numdet],xr,yr,round(image5.Width/2),round(i
mage5.Height/2), xc,yc);
   // for j:=0 to kolpointdet[numdet]-1 do
  end
 else
  begin
   For i:=0 to Nob-1 do
    begin
     xrp[i]:=-xr[i];
     yrp[i]:=-yr[i];
    end;
   MaxMin(ne,Nob,xrp,xe,1);
   ForDet(Nob,ne,xrp,yrp,xn,yn);
   GodVip(Nob,Nob,xn,yn,xr,yr,xpg,ypg,npg);
   for j:=0 to npg-1 do
    begin
     xgp[j]:=xpg[j]+xg[q];
     ygp[j]:=ypg[j]+yg[q];
    end;
      For i:=0 to npg-2 do
       For j:=0 to npg-2 do
       begin
         if
PointCrossTwoLine(Xpg[i],Ypg[i],Xpg[i+1],Ypg[i+1],Xgp[j],Ygp[j],Xgp[j+1],Ygp[j
+1], x0,y0) then
```

```
          begin
           p:=p+1;
           xp[p]:=x0;
           yp[p]:=y0;
          end
        end;
     Spar:=Abs((Xp[2]-Xp[1])*Yg[q]-Xg[q]*(Yp[2]-Yp[1]))/200000000;
    for j:=0 to kolpointdet[numdet]-1 do
     begin
      xr[j]:=-xm[numdet,j]+xp[1];
      yr[j]:=-ym[numdet,j]+yp[1];
     end;

Graph_im5(1,mxy/4,kolpointdet[numdet],xr,yr,round(image5.Width/2),round(image5.
Height/2), xc,yc);
    for j:=0 to kolpointdet[numdet]-1 do
     begin
      xr[j]:=-xm[numdet,j]+xp[2];
      yr[j]:=-ym[numdet,j]+yp[2];
     end;

Graph_im5(1,mxy/4,kolpointdet[numdet],xr,yr,round(image5.Width/2),round(image5.
Height/2), xc,yc);
    // for j:=0 to kolpointdet[numdet]-1 do

   //    XcYc(ng,xg,yg,xc,yc);

  for j:=0 to kolpointdet[numdet]-1 do
      begin
       xrr[j]:=-xm[numdet,j]+xg[q]+xp[1];
       yrr[j]:=-ym[numdet,j]+yg[q]+yp[1];
      end;

        Graph_im5(1,mxy/4,kolpointdet[numdet],xrr,yrr,round(image5.Width
/2),round(image5.Height/2), xc,yc);

   for j:=0 to kolpointdet[numdet]-1 do
      begin
       xrr[j]:=-xm[numdet,j]+xg[q]+xp[2];
       yrr[j]:=-ym[numdet,j]+yg[q]+yp[2];
      end;
```

```
  //
Graph_im5(16,mxy/4,ng,xgp,ygp,round(image5.Width/2),round(image5.Height/2),
xc,yc);
       Graph_im5(1,mxy/4,kolpointdet[numdet],xrr,yrr,round(image5.Width
/2),round(image5.Height/2), xc,yc);
   For i:=1 to p do
      begin

ellipsONE5(2,xp[i],yp[i],xc,yc,round(image5.Width/2),round(image5.Height/2),mxy/4
);

ellipsONE5(2,xp[i]+xg[q],yp[i]+yg[q],xc,yc,round(image5.Width/2),round(image5.
Height/2),mxy/4);
      end ;
Graph_im5(2,mxy/4,ng,xgp,ygp,round(image5.Width/2),round(image5.Height/2),
xc,yc);
Graph_im5(2,mxy/4,ng,xpg,ypg,round(image5.Width/2),round(image5.Height/2),
xc,yc);
  Pen.Color:=ClBlue;

MoveTo(round((Xp[1]+Xc)*mxy/4+image5.Width/2),round((Yp[1]+Yc)*mxy/4+imag
e5.Height/2));

LineTo(round((Xp[2]+Xc)*mxy/4+image5.Width/2),round((Yp[2]+Yc)*mxy/4+image
5.Height/2));

LineTo(round((Xp[2]+Xg[q]+Xc)*mxy/4+image5.Width/2),round((Yp[2]+Yg[q]+Yc
)*mxy/4+image5.Height/2));

LineTo(round((Xp[1]+Xg[q]+Xc)*mxy/4+image5.Width/2),round((Yp[1]+Yg[q]+Yc
)*mxy/4+image5.Height/2));

LineTo(round((Xp[1]+Xc)*mxy/4+image5.Width/2),round((Yp[1]+Yc)*mxy/4+image
5.Height/2));

  end;
   Sdet:=0;
      for i:=0 to kolpointdet[numdet]-2  do
       Sdet:=Sdet+xm[numdet,i]*ym[numdet,i+1]-ym[numdet,i]*xm[numdet,i+1];
       Sdet:=abs(Sdet/200000000);
       Ykl:= Sdet/Spar*100;
       Font.Name:='Ariel';
```

```
      Font.Size:=10;
{      Font.Style:=[fsBold];,fsItalic];}
      Font.Color:=clBlack;
      TextOut(image5.Width-250,20,'Щільність
укладки='+FloatToStrF(Ykl,ffFixed,6,2)+'%');
      TextOut(image5.Width-250,40,'Площа
деталі='+FloatToStrF(SDet,ffFixed,6,2)+'кв.дм');
      TextOut(image5.Width-250,60,'Площа
паралелограму='+FloatToStrF(Spar,ffFixed,6,2)+'кв.дм');

 end;
end;

procedure TForm1.Button4Click(Sender: TObject);
var  nxi,ni,ne,i,j,p,q,qr,npg,ng:integer; xc,yc,xc1,yc1,xi,xe,yi,ye,x0,y0:int64;
  xr,yr,xn,yn,xg,yg,xgp,ygp,xp,yp,xpr,ypr,xrp,yrp,xpg,ypg,xrr,yrr:mas2;
  Spar,SDet,Ykl,Sp:real;
begin
   maxmin(nxi,kolpointdet[numdet],xm[numdet],xi,-1);
   ForDet(kolpointdet[numdet],nxi,xm[numdet],ym[numdet],XobR,YobR);
   VipDet(kolpointdet[numdet],XobR,Yobr,Xob,Yob,Nob);
   VipDet(Nob,Xob,Yob,Xob,Yob,Nob);
   MaxMin(ni,Nob,Xob,xi,-1);
   ForDet(Nob,ni,Xob,Yob,xr,yr);
   MaxMin(ne,Nob,xr,xe,1);
   ForDet(Nob,ne,xr,yr,xn,yn);
   GodVip(Nob,Nob,xr,yr,xn,yn,xg,yg,ng);

with image5.Canvas do
 begin
 pen.Color:=ClWhite;
 brush.Color:=ClWhite;
 brush.Style:=BsSolid;
  rectangle(0,0,image5.Width,image5.Height);
//  XcYc(ng,xg,yg,xc1,yc1);
 XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);

Graph_im5(numdet,mxy/4,kolpointdet[numdet],xm[numdet],ym[numdet],round(imag
e5.Width/2),round(image5.Height/2), xc,yc);
If RadioButton1.Checked then
 begin
  For q:=0 to ng-1 do
```

```
 begin
  p:=0;
  for j:=0 to ng-1 do
    begin
    xgp[j]:=xg[j]+xg[q];
     ygp[j]:=yg[j]+yg[q];
    end;
   for j:=0 to kolpointdet[numdet]-1 do
    begin
     xr[j]:=xm[numdet,j]+xg[q];
     yr[j]:=ym[numdet,j]+yg[q];
    end;
//     XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);
    For i:=0 to ng-2 do
     For j:=0 to ng-2 do
      begin
       if
PointCrossTwoLine(Xg[i],Yg[i],Xg[i+1],Yg[i+1],Xgp[j],Ygp[j],Xgp[j+1],Ygp[j+1],
x0,y0) then
        begin
         p:=p+1;
         xp[p]:=x0;
         yp[p]:=y0;
        end;
       end;
    Spar:=Abs((Xp[2]-Xp[1])*Yg[q]-Xg[q]*(Yp[2]-Yp[1]))/200000000;
       If q=0 then
        begin
        qr:=q;
        xpr:=Xp;
        ypr:=Yp;
        Sp:=Spar
        end
       else
        If Spar<Sp then
         begin
          qr:=q;
         xpr:=Xp;
         ypr:=Yp;
         Sp:=Spar
         end;
    end;
```

```
//   XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);
//   Graph_im1(1,mxy/4,ng,xg,yg,round(image1.Width/2),round(image1.Height/2),
xc1,yc1);
//   Graph_im1(16,mxy/4,ng,xgp,ygp,round(image1.Width/2),round(image1.Height/2),
xc1,yc1);
  for j:=0 to kolpointdet[numdet]-1 do
    begin
    xr[j]:=xm[numdet,j]+xg[qr];
     yr[j]:=ym[numdet,j]+yg[qr];
    end;

Graph_im5(numdet,mxy/4,kolpointdet[numdet],xr,yr,round(image5.Width/2),round(i
mage5.Height/2), xc,yc);
    For i:=1 to p do
     begin

ellipsONE5(2,xpr[i],ypr[i],xc,yc,round(image5.Width/2),round(image5.Height/2),mxy
/4);
     end ;
     ellipsONE5(2,0,0,xc,yc,round(image5.Width/2),round(image5.Height/2),mxy/4);

ellipsONE5(2,xg[qr],yg[qr],xc,yc,round(image5.Width/2),round(image5.Height/2),mx
y/4);

     Pen.Color:=ClBlue;
     MoveTo(round(Xc*mxy/4+image5.Width/2),round(Yc*mxy/4+image5.Height/2));

LineTo(round((Xpr[2]+Xc)*mxy/4+image5.Width/2),round((Ypr[2]+Yc)*mxy/4+ima
ge5.Height/2));

LineTo(round((Xg[qr]+Xc)*mxy/4+image5.Width/2),round((Yg[qr]+Yc)*mxy/4+ima
ge5.Height/2));

LineTo(round((Xpr[1]+Xc)*mxy/4+image5.Width/2),round((Ypr[1]+Yc)*mxy/4+ima
ge5.Height/2));
     LineTo(round(Xc*mxy/4+image5.Width/2),round(Yc*mxy/4+image5.Height/2));

     Spar:=Abs(Xpr[1]*Ypr[2]-Xpr[2]*Ypr[1])/100000000;

   for j:=0 to kolpointdet[numdet]-1 do
    begin
```

```
     xr[j]:=xm[numdet,j]+xpr[1];
     yr[j]:=ym[numdet,j]+ypr[1];
    end;

Graph_im5(numdet,mxy/4,kolpointdet[numdet],xr,yr,round(image5.Width/2),round(i
mage5.Height/2), xc,yc);
   for j:=0 to kolpointdet[numdet]-1 do
    begin
     xr[j]:=xm[numdet,j]+xpr[2];
     yr[j]:=ym[numdet,j]+ypr[2];
    end;

Graph_im5(numdet,mxy/4,kolpointdet[numdet],xr,yr,round(image5.Width/2),round(i
mage5.Height/2), xc,yc);
  // for j:=0 to kolpointdet[numdet]-1 do
  end
  else
   begin
//    qr:=1;
{    For i:=0 to Nob-1 do
    begin
     xrp[i]:=-xr[i];
     yrp[i]:=-yr[i];
    end;
   MaxMin(ne,Nob,xrp,xe,1);
   ForDet(Nob,ne,xrp,yrp,xn,yn);
   GodVip(Nob,Nob,xr,yr,xn,yn,xpg,ypg,npg);}

For i:=0 to Nob-1 do
    begin
     xrp[i]:=-xr[i];
     yrp[i]:=-yr[i];
    end;
   MaxMin(ne,Nob,xrp,xe,1);
   ForDet(Nob,ne,xrp,yrp,xn,yn);
   GodVip(Nob,Nob,xn,yn,xr,yr,xpg,ypg,npg);
{   for j:=0 to npg-1 do
    begin
    xgp[j]:=xpg[j]+xg[q];
     ygp[j]:=ypg[j]+yg[q];
    end;}
```

```
For q:=0 to ng-1 do
  begin
   p:=0;
    for j:=0 to npg-1 do
    begin
    xgp[j]:=xpg[j]+xg[q];
     ygp[j]:=ypg[j]+yg[q];
    end;
 {   for j:=0 to kolpointdet[numdet]-1 do
     begin
      xr[j]:=xm[numdet,j]+xg[q];
      yr[j]:=ym[numdet,j]+yg[q];
     end;}
//     XcYc(kolpointdet[numdet],xm[numdet],ym[numdet],xc,yc);

     For i:=0 to npg-2 do
      For j:=0 to npg-2 do
      begin
        if
PointCrossTwoLine(Xpg[i],Ypg[i],Xpg[i+1],Ypg[i+1],Xgp[j],Ygp[j],Xgp[j+1],Ygp[j
+1], x0,y0) then
         begin
         p:=p+1;
          xp[p]:=x0;
          yp[p]:=y0;
         end;
        end;

   //      Spar:=Abs(Xp[1]*Yp[2]-Xp[2]*Yp[1])/100000000;
     Spar:=Abs((Xp[2]-Xp[1])*Yg[q]-Xg[q]*(Yp[2]-Yp[1]))/200000000;

         If q=0 then
          begin
          qr:=q;
          xpr:=Xp;
          ypr:=Yp;
          Sp:=Spar
          end
         else
          If Spar<Sp then
           begin
            qr:=q;
```

```
          xpr:=Xp;
         ypr:=Yp;
         Sp:=Spar
         end;

   end;
      for j:=0 to kolpointdet[numdet]-1 do
    begin
     xr[j]:=-xm[numdet,j]+xpr[1];
     yr[j]:=-ym[numdet,j]+ypr[1];
    end;

Graph_im5(1,mxy/4,kolpointdet[numdet],xr,yr,round(image5.Width/2),round(image5.
Height/2), xc,yc);
   for j:=0 to kolpointdet[numdet]-1 do
    begin
     xr[j]:=-xm[numdet,j]+xpr[2];
     yr[j]:=-ym[numdet,j]+ypr[2];
    end;

Graph_im5(1,mxy/4,kolpointdet[numdet],xr,yr,round(image5.Width/2),round(image5.
Height/2), xc,yc);
   // for j:=0 to kolpointdet[numdet]-1 do

  //    XcYc(ng,xg,yg,xc,yc);

  for j:=0 to kolpointdet[numdet]-1 do
     begin
      xrr[j]:=-xm[numdet,j]+xg[qr]+xpr[1];
      yrr[j]:=-ym[numdet,j]+yg[qr]+ypr[1];
     end;

       Graph_im5(1,mxy/4,kolpointdet[numdet],xrr,yrr,round(image5.Width
/2),round(image5.Height/2), xc,yc);

  for j:=0 to kolpointdet[numdet]-1 do
     begin
      xrr[j]:=-xm[numdet,j]+xg[qr]+xpr[2];
      yrr[j]:=-ym[numdet,j]+yg[qr]+ypr[2];
     end;
```

```
  //
Graph_im5(16,mxy/4,ng,xgp,ygp,round(image5.Width/2),round(image5.Height/2),
xc,yc);
       Graph_im5(1,mxy/4,kolpointdet[numdet],xrr,yrr,round(image5.Width
/2),round(image5.Height/2), xc,yc);
      Pen.Color:=ClBlue;

MoveTo(round((Xpr[1]+Xc)*mxy/4+image5.Width/2),round((Ypr[1]+Yc)*mxy/4+im
age5.Height/2));

LineTo(round((Xpr[2]+Xc)*mxy/4+image5.Width/2),round((Ypr[2]+Yc)*mxy/4+ima
ge5.Height/2));

LineTo(round((Xpr[2]+Xg[qr]+Xc)*mxy/4+image5.Width/2),round((Ypr[2]+Yg[qr]
+Yc)*mxy/4+image5.Height/2));

LineTo(round((Xpr[1]+Xg[qr]+Xc)*mxy/4+image5.Width/2),round((Ypr[1]+Yg[qr]
+Yc)*mxy/4+image5.Height/2));

LineTo(round((Xpr[1]+Xc)*mxy/4+image5.Width/2),round((Ypr[1]+Yc)*mxy/4+ima
ge5.Height/2));

   end;
    Sdet:=0;
      for i:=0 to kolpointdet[numdet]-2  do
       Sdet:=Sdet+xm[numdet,i]*ym[numdet,i+1]-ym[numdet,i]*xm[numdet,i+1];
       Sdet:=abs(Sdet/200000000);
       Ykl:= Sdet/Sp*100;
       Font.Name:='Ariel';
       Font.Size:=10;
{       Font.Style:=[fsBold];,fsItalic];}
       Font.Color:=clBlack;
       TextOut(image5.Width-250,20,'Щільність
укладки='+FloatToStrF(Ykl,ffFixed,6,2)+'%');
       TextOut(image5.Width-250,40,'Площа
деталі='+FloatToStrF(SDet,ffFixed,6,2)+'кв.дм');
       TextOut(image5.Width-250,60,'Площа
паралелограму='+FloatToStrF(Sp,ffFixed,6,2)+'кв.дм');


   for j:=0 to kolpointdet[numdet]-1 do
     begin
```

```
     xr[j]:=xm[numdet,j]+xg[qr];
     yr[j]:=ym[numdet,j]+yg[qr];
    end;

Graph_im5(numdet,mxy/4,kolpointdet[numdet],xr,yr,round(image5.Width/2),round(i
mage5.Height/2), xc,yc);

 end
end;



procedure TForm1.Button2Click(Sender: TObject);
begin
 //PageControl1.Visible:=True;
 PageControl1.ActivePage:=TabSheet1;
 Open1.Visible:=True;
end;

End.
```