Державний комітет зв'язку та інформатизації України
Національна академія наук України
Державний науково-дослідний інститут інформаційної інфраструктури

На правах рукопису

Березька Катерина Миколаївна

УДК 621.397.3

МОДЕЛЮВАННЯ ТА СИНТЕЗ СКЛАДНИХ ЗОБРАЖЕНЬ СИМЕТРИЧНОЇ СТРУКТУРИ

01.05.02 – математичне моделювання та обчислювальні методи

Дисертація
на здобуття наукового ступеня кандидата технічних наук

Науковий керівник:
Грицик Володимир Володимирович
Член кореспондент НАН України,
доктор технічних наук, професор

Львів-1999

ЗМІСТ

ВСТУП

Актуальність теми. Класи зображень, наділених симетричною структурою широко представлені в природі, мистецтві та інших галузях людського життя. Важливе дослідження класів складних зображень, наділених симетричною структурою, належить наукам: математиці, фізиці, кристалографії, хімії, біології та ін. [1-7]. Особливо широкі класи таких зображень зустрічаються в науково-технічних розробках при побудові систем штучного інтелекту, в представленні, описі, обробці та розпізнаванні образів [8-14].

Ці класи зображень відображають великі обсяги різних природних об'єктів та реалізованих фізичних і технічних процесів - одновимірних, двовимірних і трьохвимірних. Прагнення якомога ширше охопити вивчення зображень, наділених складною структурою, привело до створення різних підходів, теорій опису, представлення, моделювання, синтезу різних класів об'єктів та процесів в науково-технічних розробках, пов'язаних з розпізнаванням образів та побудовою систем штучного інтелекту [8-10]. Незважаючи на великий інтерес до моделювання та синтезу складних зображень у теорії розпізнаванні образів, мало було приділено уваги питанням симетричних структур. Власне в цій галузі людських знань є певна прогалина у дослідженні надзвичай широкого класу складних зображень, наділених симетричною структурою.

У представленій дисертаційній роботі здійснена спроба заповнити цю прогалину, розв'язуючи задачу опису, моделювання та синтезу складних зображень наділених симетричною структурою. Предметною областю вибрані зображення-орнаменти (вишивки), які представляють собою складну структуру. Цей підхід, що пропонується для розв'язку задачі моделювання та синтезу складних зображень-орнаментів, представляє великий інтерес не тільки у розвитку української культури, мистецтва, а він має також безпосередньо важливий практичний інтерес для науково-технічних розробок у побудові систем штучного інтелекту та структур обробки і розпізнавання образів. Класи зображень-орнаментів - це не тільки

вишивки у мистецтві і культурі народу, а це, на наш погляд, новий і дуже широкий клас зображень, наділених симетричною структурою. Ми називаємо цей клас складних зображень класом зображень-орнаментів або вишивками. Ці класи зображень-орнаментів відображають великі обсяги природніх симетричних зображень різного походження і штучно реалізованих. Симетрія - це фундаментальна особливість природи і вона охоплює всі форми руху і організацію матерії [1].

На сучасному етапі одним із актуальних напрямів розвитку інформаційних технологій є машинна або комп'ютерна графіка (КГ) [15-21]. Широке розповсюдження КГ зумовлене необхідністю в переробці, аналізі і відображенні візуальної інформації. Серед відомих методів представлення зображень можна виділити наступні [22, 23]:

* пряме (матричне) - найбільш просте, але зв'язане з великими затратами на обробку відеоданих;
* опис зображення за допомогою коефіцієнтів ортогонального перетворення - перетворення Фур'є, Адамара (неефективно проводяться локальні операції над зображеннями) ;
* пірамідально-рекурсивне представлення [14] - зображення описується впорядкованою послідовністю декількох зображень різного розрізнення (виникають технічні труднощі при аналізі складних зображень);
* синтаксичні (структурні) методи представлення, які базуються на описі зображень складених об'єктів у вигляді ієрархічної структури [24, 25]. Складений об'єкт зображення описується за допомогою виділеного набору непохідних елементів та правил їх з'єднання. Однак при використанні даних методів виникають проблеми вибору непохідних елементів.

Найбільш прийнятним для опису складних симетричних зображень є структурний метод, який дозволяє описувати зображення шляхом виділення мінімального рисунку та його перетворення за допомогою симетричних груп.

Всі наявні програмні засоби КГ поділяються на наступні групи [17]: програмне забезпечення для зовнішніх пристроїв, графічні компоненти операційних

систем, графічні складові мов програмування, графічні бібліотеки та пакети прикладних програм, графічні редактори. Особливе місце серед даних груп займають графічні редактори, які надають користувачеві всі необхідні програмні засоби та інструменти для створення зображень. Наявні графічні редактори поділяються на універсальні та спеціалізовані. Універсальні графічні редактори мають вартісно-функціональну надлишковість як програмного так і апаратного забезпечення. Спеціалізовані редактори призначені для вузької сфери використання (одна задача або клас подібних задач), мають функціональні обмеження, що не дозволяє використати їх як універсальну систему. Переваги і недоліки універсальних і спеціалізованих редакторів наведені у табл. 0.1.

Таблиця 0.1

Особливості графічних редакторів

Редактор	**Переваги**	**Недоліки**
Універсальний	широкий набір прикладних програм і систем проектування; мінімум затрат на створення і редагування зображень	функціонально-вартісна надлишковість, недостатність спеціалізованих цільових функцій.
Спеціалізований	оптимальне співвідношення цільових функцій до затрат; кращі цільові технічні характеристики.	розробка систем “з нуля”; складність адаптації і компонування (відсутність відкритої інформаційної взаємодії); складність розробки.

Надлишковість універсальних редакторів компенсується зниженням затрат на впровадження за рахунок скорочення часу розробки високооплачуваними спеціалістами. Такий підхід на Заході використовується і для побудови спеціалізованих редакторів, однак для України він знайшов обмежене використання через низьку купівельну спроможність замовника. Так, для IBM PC ціна ОС Windows’98 становить $260 , а остання (дев’ята) версія Corel DRAW (фірма Corel) $495 [26, 27], тобто лише вартість універсального апаратного і ліцензійного

програмного забезпечення перевищує вартість типової розробки спеціалізованих редакторів на Україні.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконана в рамках Державної науково-технічної програми 6.02.02 з пріоритетного напрямку «Перспективні інформаційні технології і системи», затвердженої ДКНТ України (1992 р.), наступних науково-дослідних тем: «Дослідження можливостей розробки високоефективних моделей і алгоритмів обробки і класифікації одномірних і двомірних сигналів із складною структурою» (номер держ. реєстрації 81041804), «Дослідження і розробка ефективних методів і засобів виявлення і класифікації сигналів і прогнозування завад» (номер держ. реєстрації 76034505), «Розробка інформаційно-аналітичної системи супроводу інтеграції України в міжнародне співтовариство» (національне агентство з питань інформатизації при Президентові України, 1998 р.).

Мета і задачі дослідження. Метою дисертації є розробка методу опису та моделювання складних зображень-орнаментів а також створення на його основі редактора зображень-орнаментів. Для досягнення поставленої мети необхідно вирішити наступні задачі:

- розробити метод опису та моделювання складних зображень, наділених симетричною структурою;
- розробити мову опису структурних частин зображення-орнаменту;
- розробити алгоритми синтезу зображень-орнаментів;
- реалізувати алгоритми синтезу орнаментального зображення-вишивки;
- реалізувати алгоритми редагування зображень-вишивок.

Наукова новизна одержаних результатів.

- Розроблено метод опису та моделювання складних зображень-орнаментів.
- На основі запропонованого методу опису та моделювання складних зображень-орнаментів розроблено мову опису структурних частин зображення-орнаменту та досліджено її властивості.
- На основі рекурсивного підходу до груп перетворень зображень, розроблено математичні моделі структурних частин зображення-орнаменту.

- Запропоновано алгоритми синтезу федорівських груп перетворень зображень на основі осьових симетрій.
- Розроблено математичні моделі перетворень зображень для федорівських груп на площині та смузі в матричному вигляді, які лягли в основу синтезу алгоритмів генерування зображень-орнаментів.
- Отримано аналітичні вирази для коефіцієнтів стиснення (архівування) зображень-орнаментів, побудованих на основі запропонованого методу.

Практичне значення одержаних результатів Практична цінність отриманих результатів полягає у тому, що: розроблені алгоритми опису, синтезу та архівування зображень-орнаментів дають можливість моделювати та автоматично генерувати складні симетричні зображення та ефективно зберігати складові зображення-орнаменту.

Запропоновані алгоритми реалізовані в наступних системах:

1) розроблено і впроваджено редактор зображень-орнаментів;
2) розроблено і впроваджено редактор вишивок.

Особистий внесок здобувача. Усі теоретичні дослідження, розробка алгоритмічного і програмного забезпечення, основна частина практичних розробок виконані автором самостійно. В друкованих працях, які приведені в співавторстві, автору належить метод опису складних зображень, алгебра мови опису та дослідження її властивостей [28, 29], алгоритми генерування груп перетворень зображень [30, 31, 32], участь в постановці задач, розробленні та програмуванні редакторів орнаментів і вишивок [33].

Апробація результатів дисертації. Результати дисертаційної роботи доповідалися на 4-ій українській конференції «Автоматика 97» (Черкаси, 1997), конференціях “Застосування обчислювальної техніки, математичного моделювання та математичних методів у наукових дослідженнях” (Львів, 1997, 1999), міжнародній науковій конференції «Сучасні проблеми математики». (Чернівці, 1998), міжнародній науковій конференції «Розробка та застосування математичних методів у науково-технічних дослідженнях» (Львів, 1998).

Публікації. За результатами виконаних досліджень опубліковано 12 робіт, із яких 7 статей у фахових наукових виданнях, 5 - матеріали і тези конференцій, із них 6 - одноосібні публікації.

РОЗДІЛ 1
АНАЛІЗ МЕТОДІВ ПРЕДСТАВЛЕННЯ СКЛАДНИХ ЗОБРАЖЕНЬ

1.1. Симетрія в природі, науці, мистецтві, архітектурі

В роботі досліджуються складні зображення, наділені симетричною структурою. Одним з класів таких зображень є орнаменти. Розглянемо поняття симетрії в різних сферах суспільного життя. Адже симетрія - це така особливість природи, яка являється фундаментальною. Вона «охоплює всі форми руху і організації матерії» [1, с.3].

На протязі всієї історії людства симетрія відігравала важливу роль в мистецтві, науці і інших видах інтелектуальної діяльності. Початок розвитку теорії симетрії поклали древні греки, вважаючи, що матеріальний світ симетричний. Вони відкрили п'ять правильних тіл і рахували, що вони є елементарними частинками з яких побудований світ [34]. З кубом вони зв'язували світ, з октаедром - повітря, з тетраедром - вогонь, з ікосаедром - воду і з додекаедром - космос.

Найчастіше в побуті слово симетрія вживається в 2-х значеннях - широкому і вузькому. Вважається, по-перше, що симетричне це те, що володіє "хорошим співвідношенням пропорцій, врівноважене, а симетрія означає той вид узгодженості окремих частин, який об'єднує їх в єдине ціле"[2]. Отже, в першому значенні, симетрія - це краса, гармонія природи. В другому значенні, симетрія - це строго геометричне поняття, а не розпливчате. «Симетрія в широкому або вузькому розумінні... є тією ідеєю, за допомогою якої людина протягом віків намагалася осягнути і створити порядок, красу і досконалість» [2].

Опис симетрії не буде повним без введення понять класичної симетрії і її узагальнень [35, 36].

Відображення (симетрія відносно прямої, дзеркальна симетрія) - різновидність симетрії, яка найчастіше зустрічається в природі. Дзеркальною симетрією володіє все, що допускає розбиття на дві дзеркально рівні частинки. Кожна з половинок є дзеркальним відображенням іншої, а уявна площина, яка розділяє їх називається

площиною дзеркального відображення. Уявне дзеркало може відображати з обох сторін, воно прозоре і не має товщини. В теорії симетрії абстрактне "дзеркало" називається елементом симетрії, а відображення в площині - операцією симетрії.

Другою різновидністю симетрії є поворот. Розміри і форма будь-якої фігури не зміняться, а всі точки будуть рухатися по дугах концентричних кіл, якщо її повернути на певний кут навколо осі, яка є перпендикулярною площині фігури. Щоб здійснити повний оберт на 360°, необхідно здійснити один за одним чотири повороти на 90°, три повороти на 120°, або шість поворотів на 60°. В цих випадках вісь називається віссю симетрії 4-го порядку, віссю симетрії 3-го порядку, віссю симетрії 6-го порядку відповідно.

Калейдоскоп - це прибор, в якому симетрія створюється за рахунок оптичного відображення довільного мотиву. Дзеркала розташовуються під деяким кутом і за рахунок цього утворюється конфігурація, яка володіє дзеркальною і поворотною симетрією. Відображення відображення співпадає з простим поворотом вихідного мотиву. Вісь повороту в калейдоскопі співпадає з лінією перетину дзеркал.

Третім видом симетрії є паралельний перенос (трансляція). Цією симетрією володіють узори, які повторюються необмежено на площині через певну відстань. Паралельний перенос - перетворення, яке зберігає відстань між будь-якими точками і напрямок прямої, що проходить через них. Якщо говорять, що відрізок A'B' є образом відрізка AB при паралельному переносі, то або точки A,B,A',B' лежать на прямій або AA'B'B - паралелограм. Трансляція в одному напрямку (разом з трансляцією в протилежному напрямку) породжує одномірний узор. Трансляція по двох непаралельних напрямках характеризує плоский узор. В трьохмірному узорі напрямки трансляції повинні бути некомпланарні.

Четверта різновидність - ковзне відображення полягає в тому, що фігуру переносять на певну відстань вздовж деякої осі, а потім відображають відносно цієї ж осі.

Симетрія зустрічається всюди. Вона проявляється не тільки в просторі, а й на площині і на лінії. «Ці правильності більш глибокі ніж фізичні і хімічні явища, в яких вони нам проявляються і які вони охоплюють» [3]. Симетрія зачіпає не тільки

те що ми бачимо дуже добре, це рослини, тварини, меблі, будова людського тіла, а й чудові і не менш важливі різновидності симетрії, які зустрічаються в літературі, образотворчому мистецтві, музиці, танці.

Симетрія виражається яскраво в орнаментах. Вона служить основним прийомом при їх побудові. З допомогою орнаменту акцентується увага на предмет, на який нанесено орнамент [37, 38].

Мистецтво орнаменту бере свій початок в стародавніх епохах людства, його зачатки зафіксовані вже в період палеоліту [39]. В цей період симетрія існує у вигляді орнаментальних повторів з зигзагів, циклоїд, синусоїд. Припускається, що синхронна правильність прикрашування була способом художнього осмислення людиною оточуючого його світу і себе в ньому.

В період неоліту орнаментальна симетрія зустрічається в барвистіших, різноманітніших геометричних формах. Тут і круги, багатокутники, зірки, спіралі і ін.. З часом геометричний орнамент змінюється на рослинний і зооморфний, які більш ближче оточують людину. Зустрічаються також людські постаті, знаряддя праці, зброя, музичні інструменти, різні знаки. Ці три характеристики комбінують в усіх наступних орнаментах багатьох цивілізацій. Але будь-який орнамент будується з закінчених композицій, які мають свій внутрішній простір [40]. "Зверху" він сприймається, як неділимо ціла частинка, як одиниця ритмічного відрахунку.

Нові епохи створили свої стилі, наприклад: єгипетський, грецький, римський, арабський, мавританський, готичний, барокко та ін. Кожен з них характеризується своїми правилами. Істинним музеєм мавританського орнаменту є Альгамбра (Гренада, Іспанія). Грецький орнамент відзначається строгою математичною стилізованістю, упорядкованістю. Закони симетрії і ритму супроводжують його. Найчастіше в ньому зустрічаються рослинні мотиви ліандрів, пальмет, окантовані листи і т.д. Різноманітні узори існують на саркофагах єгипетських фараонів.

На основі геометрії при ефективному використанні архітекторами в своїх творах закономірностей симетрії вирішуються функціональні і естетичні питання архітектурної форми. В архітектурі переважно зустрічаються дзеркальна і переносна симетрія. Часто буває осьова симетрія четвертого, п’ятого і ін. порядків. Наприклад

[41], собор св. Марії дельї Анджели в Флоренції (XV ст.) має форму восьмикутника, тобто симетрію 8-го порядку. Симетрія 6-го порядку найчастіше зустрічається в такому елементі споруд як башні. Архітектори втілюють симетрію і в фасадах споруд і в розбивці планів на місцевості.

Леонардо да Вінчі багато займався питаннями симетрії в архітектурі. Він визначав можливі види симетрії як для центральних споруд так і добудованих, щоб не порушувалась симетрія архітектурного ансамблю. Його результати співпали з можливими скінченними групами повороту, які привів Г. Вейль - власними і дзеркальними [2, 41] (підрозділ 2.1).

Симетрія проявляється на тільки в зовнішньому вигляді споруд, а й всередині їх, а саме в настінних розписах. «...Однією з характерних стильових рис ... є конструктивність, визначена геометричність, посилена послідовним дотримуванням принципу симетрії» [42, с.54]. За допомогою розписів стін творці прагнули передати природу як поєднання добрих і злих сил (стихій). В них втілено «живе сприйняття навколишньої дійсності». Українські народні настінні розписи першої третини ХХст. - це були орнаменти, що будувалися з «вазонів», що розміщувалися здебільшого по принципу дзеркальної симетрії. Розписи створювалися від руки, тому інколи симетрія порушувалася, що було видно на око.

Симетрія - життєво важлива складова частина будь-якого аспекту музики. Вона досить часто зустрічається в симфоніях і в мелодійних творах, а особливо в дитячих пісеньках, може для того, щоб вони краще запам'ятовувались. Вона проявляється найчастіше в повторах (паралельних переносах) і відображеннях, але повтори і відображення проходять не в просторі, а в часі. Симетрія породжує порядок і подібно орнаментам створює свій рисунок з музичних елементів.

В епоху класицизму і в часи Баха було розповсюджено симетричне розташування частин в музичному творі. На той час симетрія була одним з основних принципів композиції в мистецтві. Дуже цікаву гру в музичні головоломки пропонував своїм слухачам геніальний композитор І.С.Бах [34]. Метою кожної з головоломок було розгадування "загадкових канонів" (форм багатоголосої музики, яка полягала в проведенні теми, яку веде один голос (пропоста), в інших голосах

(риспостах). Тема пропонувалася самим композитором, а гравцям необхідно було відгадати операції симетрії, які він планував використати при повторі теми.

Симетрія спостерігається і в танцях. Кожний танець має свою особливу симетрію, лише в окремих випадках в танці проходить перебудова, яка порушує симетрію. Симетрія в танцях полягає в кількості виконавців і фігурах танців.

Симетрію бачимо і в тонких роботах ювелірних майстрів, в жіночих прикрасах.

Симетрія характерна українській народній вишивці. Інколи вона порушується за рахунок використання ниток різного кольору, але основа є симетричною [43-45].

Симетрія проявляється в літературі. Висока точність, яку вимагає симетрія, в літературі зустрічається дуже рідко. В літературі можна побачити лише тягу до симетрії і деякі симетричні ефекти. Повороти, відображення чи трансляції в літературних творах не слід розуміти буквально. Симетрія включає в себе мислення і спонукальні мотиви персонажів в рамках авторської розповіді. Найчастіше вона проявляється синтетично, як модель при побудові твору (в сюжетах по спіралі). Різні форми рядків і віршованих творів володіють наближеною симетрією. Найчастіше це відноситься до приспівів в віршованих текстів пісень. При цьому ці повтори надають певної естетичної насолоди.

В усному фольклорі симетрія проявляється найчастіше в казках, в яких декілька раз повторюються ті ж самі дії чи елементи розповіді.

Симетрія зустрічається в ілюстрованих книгах. В епоху зародження поліграфії і книгодрукування прикрашали книги з допомогою відлитих декоративних деталей орнаментів. Їх розміщували в різних послідовностях, накладали фарбу і, таким чином, одержували на папері відтиски орнаменту. Найчастіше декоративними деталями були арабески - складні узори, які побудовані з складних переплетень геометричних і стилізованих рослинних мотивів. Назва "арабеска" означає, що узор виконано в арабському стилі. З часом так почали називатися всі причудливі, складні, багаті узори.

Типографський орнамент досяг свого розквіту в Франції в середині XVI ст. Крихітні вишукані творіння (арабески) допомогли забезпечити країні провідне місце в книгодрукуванні епохи Відродження.

На Україні орнаментальне оформлення друкованої книги пов'язане з відкриттям Іваном Федоровим друкарні у другій половині XVI ст. Орнаментальне оформлення книги здійснювалося рослинною орнаментикою ("Апостол" 1574 р.; П.Беринда "Євангеліє Учительне" 1606 р.; "Служебник" Йова Борецького 1629 р.) [46]. А рукописні заставки книги "Ірмологіон" кінця XVII ст. та "Літопису Софоновича" 1681 р. свідчать про те, що орнаменти книг нерозривно взаємодіють з народним мистецтвом. Квіти виконано чорним штрихом, графічно чітко, симетрично. Внутрішня штрихова розробка наче імітує стібки ниток при вишиванні.

Симетрія присутня в багатьох областях природознавства і пояснюється взаємозв'язком симетрії причини і наслідку. Часто необхідно знайти причину, що викликає той чи інший ефект і передбачити ефект викликаний тою чи іншою причиною. Симетрія ефекту свідчить про закономірний характер першопричини явища або структури системи. Сама симетрія в зв'язку з її широким застосуванням стала цінним об'єктом дослідження. Тому що операції симетрії мають особливе значення як причини поскільки викликані ними ефекти повністю передбачені. Дуальність причини і наслідку, причини і ефекту приводить до вивчення симетрії.

«...В явищах природи є форми і ритми недоступні оку спостерігача, але відкриті оку аналітика. Ці форми і ритми ми називаємо фізичними законами» ([4, стор. 9]). В фізиці, існування певної якісної збереженої характеристики явища, зумовлено симетрією розглядуваного явища, тобто інваріантністю (незмінністю) всіх рівнянь, що описують його відносно деякого перетворення величин, що входять в нього. Прикладами є закон збереження енергії, закон збереження імпульсу для замкнутої системи, закон збереження моменту імпульсу для замкнутої системи і ін. Перший зумовлюється однорідністю часу, тобто інваріантністю рівнянь руху замкнутої системи матеріальних точок відносно переносу початку відліку часу. При цьому замкнута система матеріальних точок знаходиться під дією силового поля, яке не залежить від часу. Другий - однорідністю простору, тобто інваріантністю рівнянь руху замкнутої системи відносно переносу її в іншу область простору. Третій - ізотропністю простору, тобто інваріантністю рівнянь руху відносно повороту системи як цілого [4].

Симетрія існує в будові молекул і атомів. Хоча ніхто з людей неозброєним оком їх не бачив і не побачить, але те що складається в уяві підтверджується лабораторними дослідженнями. Молекули складаються з заряджених частинок - електронів і атомних ядер. Для дослідження відносного розташування атомних ядер були розроблені точні методи: дифракція рентгенівського випромінювання, електронних і нейтронних пучків при проходженні через речовину, поглинання речовиною випромінювання і її взаємодії з магнітним полем. Кількісна інтерпретація експерементальних даних про будову молекул спирається на положення теорії симетрії. Були розроблені методи, що дозволяють кількісно передбачати розташування атомних ядер в молекулах на основі принципів квантової механіки. Передбачення розташування атомних ядер вимагають володіння теорією симетрії. Положення симетрії спрощують задачу квантової механіки. При побудові моделей молекул використовують властивості симетрії. Спільність симетрії моделі і молекули - шлях дослідження хіміків [34, 5].

В період росту і розвитку рослин біологи бачать безліч цікавих прикладів симетрії. Її яскраво видно в будові всієї рослини. Кожна рослина характеризується специфічною симетрією, що дозволяє розпізнавати її серед інших видів. Нерідко вона зустрічається в будові коренів (володіють радіальною симетрією), стебел, але найбільш яскраво виражена в таких органах як листки і квіти. Дзеркальною симетрією володіють гілки дерева, квіти, що розміщені на стеблі збоку. Поворотна симетрія спостерігається в деревах, квітах, які повернуту чашечкою вверх. А в мікроскоп людина бачить симетрію внутрішньої будови. Внутрішня будова судинних рослин має цікаву симетрію. На поперечніх січеннях тканин, які утворюють стебло або корінь рослини, присутня радіальна симетрія. Цікавий вид симетрії спостерігається у ільма американського [34], в якого листки в кожному вузлі направлені в протилежні сторони (розходяться на 180 °). Листкове розміщення утворює дроби по наступній послідовності: 1/2, 1/3, 2/5, 3/8, 5/13, 8/21, 13/34, 21/55, 34/89. Дана послідовність належить до послідовностей типу Фібоначі. В соняшнику сім’я в головках розташовані по логарифмічним спіралям, які розкручуються в протилежні сторони, і перетинаються під кутами близькими до

прямих. Поворотна симетрія 5-го порядку зустрічається в квітах, наприклад, шипшини. Яблуко в поперечному розрізі теж володіє такою ж симетрією,а квіти ірису симетрією 3-го порядку. Дзеркальною симетрією володіють метелики, риби, птахи, ссавці і люди, поворотною - нижчі тварини - медузи, морські їжаки, морські зірки і ін.

Автор [6] підмітив таку закономірність в природі. Поворотну симетрію має в основному те, що росте або рухається по вертикалі, а дзеркальну - те, що росте або рухається по горизонталі чи під кутом до земної поверхні. Свої висновки він пояснює наявністю поля земного тяжіння (гравітацією), що впливає на всі організми.

Геометричні форми кристалів завжди дивували уяву людей. Привабливість кристалів пояснюється не тільки естетичними, але й науковими поясненнями. Кристалографи, провівши тисячі скорпульозних вимірювань зовнішньої форми кристалів, знайшли важливі емпіричні "закони". Але найбільш значимою подією в кристалографії стало відкриття Гаюї, що встановлює, що для будь-якого кристалу можна вибрати систему координат, в якій рівняння площин що містять грані кристалу мають раціональні коефіцієнти. Іншими словами кристал - це раціональний багатогранник. Закон Гаюї дає можливість передбачувати періодичність внутрішньої будови кристалів. Грані кристалів зустрічаються не поодинці, а симетрично розташованими сім'ями. В кристалографії введено поняття форми - сукупність граней кристалу, кожна з яких рівнооточена іншими гранями цього кристалу. Кожну грань можна перевести в іншу грань тієї ж форми за допомогою операції симетрії, тобто з кожною формою зв'язана деяка група симетрії. Різні форми, що спостерігаються на окремому кристалі, можуть володіти різними групами симетрії. Група симетрії кристалу - найбільша група, яка одночасно відображає кожну форму на себе. Групи симетрії, якими може володіти кристал, це групи що не володіють поворотними осями симетрії 5-го чи більше ніж 6-го порядку. Всього існує 32 кристалічних класи.

Питання про кількість типів елементів симетрії, кількість елементів симетрії в одному узорі симетрії вивчались в багатьох цивілізаціях. На початку XX ст. було

відкрито, що узори поділяються на скінченну кількість типів. Отже, між узорами всіх областей мистецтва, природознавства існують фундаментальні співвідношення. Найвагоміший внесок в дані дослідженння було зроблено видатним російським вченим Є.С. Федоровим [4, 7]. Він дослідив не тільки кристалографічні групи в просторі про які мова йшла вище, а й періодичні орнаменти на площині і на смузі.

Для площин все багатство періодичних узорів вклалося в 17 груп, для смуги - в 7. Приведемо їх позначення і породжувальні перетворення (позначення прийняті згідно з міжнародною системою позначень плоских кристалографічних груп [47]). Групи площини наступні:

p1 - група, що одержується двома паралельними переносами;

p2 - група, що одержується трьома центральними симетріями;

pm - група, що одержується двома осьовими симетріями і паралельним переносом;

pg - група, що одержується двома ковзними симетріями з паралельними осями;

cm - група, що одержується осьовою і сковзною симетрією з паралельними осями;

pmm - група, що одержується симетрією відносно 4-ох сторін прямокутника;

pmg - група, що одержується однією осьовою і двома центральними симетріями;

pgg - група, що одержується двома ковзними симетріями з перпендикулярними осями;

cmm - група, що одержується двома осьовими симетріями з перпендикулярними осями і однією центральною;

p4 - група, що одержується центральною симетрією і поворотом на 90°;

p4m - група, що одержується симетрією відносно 3-х сторін прямокутного рівнобедреного трикутника;

p4g - група, що одержується осьовою симетрією і поворотом на 90°;

p3 - група, що одержується двома поворотами на 120°;

p31m - група, що одержується осьовою симетрією і поворотом на 120°;

p3m1 - група, що одержується симетрією відносно трьох сторін рівностороннього трикутника;

p6 - група, що одержується центральною симетрією і поворотом на 120°;

p6m - група, що одержується симетрією відносно трьох сторін прямокутного

трикутника з кутом 30°.

Приведемо позначення і породжувальні перетворення для груп смуги. Вони будуть наступні:

p1 - група, що одержується тільки паралельним переносом;

pg - група, яка одержується ковзною симетрією (перенос на відрізок *a* з осьовою симетрією відносно осі);

p1m - група, яка одержується двома осьовими симетріями;

p2 - група, яка одержується двома центральними симетріями (напівоборотами);

pmg - група, яка одержується 1-ю осьовою і 1-ю центральною симетрією;

pm - група, яка одержується 1 паралельним переносом і 1- єю осьовою симетрією;

pmm - група, яка одержується трьома осьовими симетріями.

Отже, в даному параграфі показано фундаментальну роль поняття симетрії (в науці, техніці, мистецтві, побуті), розкрито її геометричну суть та приведено групи перетворень симетричних зображень для площини та смуги. З проаналізованих джерел видно, що питання моделювання симетричних структур не знайшли свого вирішення, тому в наступному розділі його буде розглянено.

1.2. Аналіз предметної області (української народної вишивки)

Як показано в підрозділі 1.1 вишивка – наділена симетричною структурою. Вишивка на Україні є одним з найулюбленіших і найрозповсюджених видів декоративно-прикладного мистецтва. В ній невичерпне багатство могутніх творчих сил народу. В творчості українського народу вишивка посідає одне з перших місць.

Перші згадки про вишивання зустрічаються у стародавніх істориків, в археологічних знахідках з скіфських поховань. До найцінніших скарбів відносяться золота пектораль з Товстої могили (IV ст. до н.е.), чаша з Гайманової могили, срібна ваза з кургану Чортомлик (IV ст. до н.е.) [44]. На них розміщені сюжети, з допомогою яких дослідники простежують декоративне оздоблення жіночого і чоловічого одягу. Для нього характерне поєднання різноманітних геометричних

фігур - трикутників, кругів, ламаних, спіралей і рослинний орнамент - сполучення стебел рослин, листочків, грон винограду, лотоса. Про вишивки згадується в історичних творах. Так, при розгромі Путивля (1147 рік) були знищені речі, вишиті золотом, а під 1164 роком записано: «и присла царь многы дары Ростиславу, оксамоты и паволокы и вся узорчя различная»[48].

Як свідчать історичні джерела, в Київській Русі заняття вишиванням були досить поширеними і престижними. Про це свідчать факти, що вишивала княгиня Анна - дружина київського князя Рюрика Ростиславича, а наприкінці XI ст. Анна-Янка - дочка великого князя Всеволода, сестра Володимира Мономаха - заснувала в Андріївському монастирі у Києві школу, в якій навчали молодих дівчат вишивати золотом і сріблом.

Мистецтво вишивання для князівського двору, церкви відрізнялося від мистецтва трудового народу. Перше живилося від культури Візантії, але воно не було лише точною копією, а скоріше осягненням художнього зразка, запозиченого з Візантії. Друге ж ввібрало традиції, звичаї, смаки попередніх поколінь, корені його в джерелах праслов'янських вірувань. Ці два напрямки в давньоруському мистецтві з часом з'єдналися. Візантійське мистецтво завдяки народній творчості стимулювало розквіт вишивального мистецтва.

XI-XIII ст. характеризувалися розквітом українського гаптування. Це було дороге шитво. Воно виконувалося золотими і срібними нитками на атласі, парчі, західноєвропейському оксамиті, китайському та перському шовку. Інколи використовували при цьому перли та коштовне каміння. Золотна орнаментальна вишивка займала провідне місце в художній культурі домонгольської Русі. Розвиток золотного гаптування був загальмований у XIII ст. татаро-монгольською навалою, але художні традиції давньоруського мистецтва не були втрачені. Найчастіше гаптовані вироби - це одяг священнослужителів та предмети, що застосовувались у літургії православного храму - фелони, плащаниці, покрівці і т. д. Гаптування цього часу розвивалось у стилістично-художній єдності з іконописом та гравюрою.

Цінним скарбом українського гаптування є золочівський фелон, який зберігається в Національному музеї у Львові (в історії мистецтва він датований як

твір XIII - початку XIV ст.) [44]. Залишились в прекрасному вигляді зображення 11 постатей цього шедевру. Роботи виконані надзвичайно вишукано, тонко змодельовані обличчя. Загальне враження гармонійності, декоративності поверхні фелону, його мерехтливість, випромінюваність підсилюється з допомогою симетричної техніки вишивання одягу, вишуканого зіставлення золотних і срібних ниток, поєднання шовку різних кольорів - пурпурного, яскравоблакитного, синьо-чорного. Симетричною технікою викладена і підлога («ялинкою») під образом Ісуса Христа, Богородиці. Інший симетричний орнамент спостерігаємо на книжці, на німбу. Техніки виконання: «у прикріп», «у ялиночку» та ін.

XVI-XVIII ст. є добою дальшого розквіту орнаментальних форм, в яких органічно поєднуються як давньоруські, так і народні традиції, переплетені із західноєвропейськими і східними мистецтвами. Вишивки виконувались переважно на домотканих матеріалах: льняних, конопляних, вовняних, а також тканинах вітчизняного мануфактурного виготовлення, шкірі місцевої виправки.

До нашого часу твори української народної вишивки дійшли лише з кінця XVIII - початку XIX ст. Це відбулося тому, що цікавитися творами вишивки як творами мистецтва прогресивні діячі культури, художники починають з 80-90-х років минулого століття. З того часу її починають колекціонувати в музеях і приватних зібраннях, що відкриваються в ряді міст України. Це Музей художньої промисловості (1874) та Етнографічний музей при науковому товаристві ім. Шевченка (1895) у м.Львові. Ініціаторами відкриття музеїв були І. Франко, В.Шухевич, М. Зубрицький, Б. Заклинський, Л. Гарматій та ін. Фонди музеїв Львова містили не тільки місцеві колекції, а й поповнювались цінними екземплярами з центральних районів України. Цьому сприяли видатні культурні діячі Леся Українка, М.Біляшівський, О. Сластіон [43]. Організовувались приватні та земські артілі, школи майстрів. Наставниками, вчителями були талановиті народні майстри, роботи яких потрапляли на всеросійські і міжнародні виставки.

Початок вивченню української вишивки вважається покладеним матір'ю видатної поетеси Лесі Українки - О.Косач (О.Пчілкою) - книгою «Український народний орнамент», що вийшла з друку в 1876 році [44]. Авторка проаналізувала

вишивки районів Волинського Полісся.

У процесі соціально-економічного і культурного розвитку на Україні сформувалися специфічні особливості народної вишивки кожного етнографічного регіону. Це проявляється в певних орнаментальних мотивах і композиціях, в своєрідній колірній гамі, в улюблених техніках виконання. Вишивки передавались з покоління в покоління, від матері до дочки, покращувалися і відшліфовувалися в порівнянні з попередніми роботами.

Для всіх регіонів характерною є наявність рослинного і геометричного орнаментів. Геометричні узори - ромб, розетка, хрестоподібні, зірковидні фігури стали основою східнослов'янського узору [44].

Академік Рибаков Б.О. займався дослідженням семантики геометричного орнаменту. Він довів, що геометричний орнамент має міфологічну основу, тісно пов'язаний з язическими віруваннями слов'ян і його корені сягають палеоліту. Він вважає, що значення знаків у вигляді 4-х, 6-и, 8-ми пелюсткової розетки в колі, їх поширення в різьбленні хат, прялок, гуцульських та полтавських скринь, порохівниць дають підставу вважати, що це графічні символи сонця. Шестипроменевий знак - «громовий знак» або «колесо Юпітера», нагадує авторові язическое небесне божество Род [39].

Архаїчні мотиви в українській орнаментиці зустрічаються в подільських рушниках, в вишивці Чернігівщини, Полтавщини у вигляді зображень фантастичних диво-птахів, грифонів, сиринів, жіночих фігур з трикутним одягом, розширеним донизу, піднятими догори руками, головою, «увінчаною обабіч зображенням птахів» [44]. Зірки в геометричному орнаменті - це уявлення про структуру Всесвіту, що є не хаотичним, а упорядкованим. Рослинні мотиви теж мають міфологічну основу. Калина - це дерево українського роду. У давні часи її пов'язували з народженням Всесвіту, вогненної трійці - Сонця, Місяця і Зорі. Дуб і калина уособлюють в собі поєднання незвичайної сили і краси.

Часто в орнаментах зустрічаються птахи. Пава - це жар-птиця, в ній «сонячна енергія розвитку, тому вона птах сімейного щастя». Силуети голубів - знак любові і парування.

Вишивка як класичний вид українського народного орнаментального мистецтва набула масового поширення завдяки тому, що людина завжди прагнула красиво прикрашати свій одяг (народні костюми - вишиті сорочки, сукні, пояси, верхній одяг, головні убори і ін.), особливо вишукано підходила до оформлення інтер'єру, до підготовки до свята [49, 50]. Кожна місцевість має свої улюблені техніки, улюблені колірні гами.

В народній вишивці пройшли вирішальні зміни завдяки появі нової техніки - вишивки хрестиком. Хрестик відомий в Європі з кінця XVIII ст., а в Росії і Україні з початку XIX ст. [43]. Цією технікою спочатку вишивали лише в містах та панських маєтках різні інтер'єрні прикраси - панно, серветки, скатерки. Поступово хрестиком почали вишивати широкі маси. Хрестик витіснив давні засоби шитва, вніс зміни і в колористичну гаму, став однією з найпоширеніших технік вишивання.

Умовно здійснено етнографічне районування території. Це Середнє Подніпров'я, Полісся, Поділля, Південь, Карпати і Прикарпаття [44]. Проведено воно згідно локальних відмінностей між специфічними техніками виконання, типовими орнаментальними мотивами і композиціями, поширеною колірною гамою.

До Середнього Подніпров'я належать райони, які розташовані по середній течії Дніпра. За сучасним адміністративним поділом входять Київська, Полтавська, Чернігівська, Житомирська, Сумська, Черкаська, Харківська, Кіровоградська, Луганська області .

Серед інших вишивок цього району виділяються полтавські. Вони характеризуються м'якими ніжними кольорами. Вміло, зі смаком поєднуються всі відтінки голубого, зеленого, сірого, пастельного. Дуже поширеною в XIX ст. була і залишається вишивка білим по білому («біллю»). Вона моделює світлотіньовий малюнок високого рельєфу. На світлі візерунок гармонійно виграє ніжною гамою кольорів. Для полтавських вишивок характерні різні типи орнаментів - рослинний, рослинно-геометричний, геометричний. В рослинних поширені мотиви «барвінок», «хмелик», «курячий брід», «зозулька», «гілки», «ламане дерево». Геометричні мотиви - це скісний та прямий хрест, квадрат, ромб, трикутник, зірчасті мотиви. Вони створюють різноманітні композиційні побудови за рахунок комбінацій

з’єднань. Характерною особливістю цього виду вишивок є «віртуозність і ювелірність внутрішньої розробки мотивів у поєднанні з чіткою лінією, що об’єднує всю композицію, створюючи вивірений спокійно-розмірений ритм» [44]. Всі геометричні орнаменти здаються неперервними за рахунок того, що одні елементи є одночасно частинами інших.

Поширеними мотивами для вишивки Київщини є грона винограду, гілочки з квітами, ягідками, «зірочки» з червоно-чорним поєднанням кольорів.

Для вишивки Чернігівщини характерні орнаменти виконані білими з невеликими добавленнями червоного і чорного кольорів. В основу рослинно-геометричного орнаменту покладено поєднання вертикальних ліній чернігівської «берізки». Також зустрічаються орнаменти з грон винограду, ягід, маленьких ніжних квіточок.

До етнографічного району Полісся відносяться території сучасних Волинської, Рівненської, Житомирської, північні райони Київської, Чернігівської, Сумської областей. В порівнянні з іншими районами України тут набагато довше збереглися в народному мистецтві, в побуті архаїчні мотиви, що несли прадавні традиції.

Геометричний орнамент Рівненської і Волинської областей складається з ритмічних повторень вписаних один в одного або поодиноких мотивів ромбів, ламаних ліній, 8-микутних зірок. В рослинному орнаменті поширені мотиви «шашечки», «ключики», «терен», «виноград».

Поділля охоплює територію між південним Бугом і Дністром. Його поділяють на Західне, Східне Поділля і Подністров’я. Подільська вишивка є найскладнішою і найкрасивішою на Україні.

Геометричні мотиви вишивки Східного Поділля нагадують дорогоцінну мозаїку (настільки складними і мініатюрними є елементи, що її утворюють). Відомими на весь світ стали вишивки жителів сіл Клембівка і Городівка, що на Вінничині. Тут вишивали не тільки для своїх потреб, а й для промислу. Вишивали тут навіть чоловіки і діти. Вироби везли для продажу на ярмарки Києва, Петербурга, Парижу, інших міст Європи. Найчастіше вироби вишивали низзю чорного і червоного кольорів. Ця техніка виконується з виворотньої сторони, а на лиці має протилежне

розташування кольорів. Щоб узори не були одноманітними, то використовували кольори червоні і чорні в шаховому порядку. Ще однією характерною особливістю вишивок села Клембівки були поєднання чорного і житнього кольорів. Це поєднання кольорів найчастіше зустрічається в вишитих сорочках і виглядає досить багато. На відміну від полтавських вишивок вишивка білим по білому характеризується «компактністю розроблених мотивів, застосуванням філігранних технік». Поширеними мотивами є «солов'їні вічка» (поєднання чотирьох невеличких квадратів з круглою дірочкою посередині), «зерновий вивід», «довбанка».

Вишивки Західного Поділля відзначаються темним колоритом. Найчастіше він спостерігається в вишитих сорочках. Так сорочки в Тернопільській області вишивалися вовною густо без пробілів, орнаменти гаптувалися суцільно. З появою техніки хрестика вишивка стала яскравою, поширилися квіткові орнаменти.

Буковинська вишивка ввібрала багато технік виконання. Це - низь, настил, хрестик, мережка. Жителі цього району основну увагу приділяють оздобленню народного костюму. Найпоширеніші геометричні мотиви - поєднання різноманітних за формою ромбів.

До території Покуття відносяться Снятинський, Городенківський, Коломийський, частина Тлумацького району Івано-Франківської області. Вишивка цього району нагадує подільську. Здебільшого зустрічаються в орнаментиці геометричні мотиви з ламаних, розміщених вертикальними смугами або в шаховому порядку.

До етнографічного району Прикарпаття і Карпат належать північне Прикарпаття - Львівщина і частина Тернопільської, гірські райони Карпат - Івано-Франківська, частина Львівської, Закарпаття.

Виділяються вишивки Львівської області. Сокальські сорочки вишиті переважно чорним кольором, технікою - хрестик або стебнівка, нагадують тонке мереживо. Для Городоцького району улюбленими є поєднання червоного з додаванням синього, інколи червоного кольорів («городоцький шов»). Найулюбленіший мотив - багатопроменева розетка. Зустрічаються також мотиви з квіток, галузок. Яворівська вишивка є дуже різноманітна. Широко популярна вишивка одягу. Для кожного виду

(кабати, камізелі, запаски, кожухи, сорочки, хустки) були свої прийоми, своя техніка виконання, певний орнамент. Поширені складні яскраві насичені квіткові композиції.

Жителі Закарпаття люблять застосовувати в вишивках бісер, стеклярус, металічні лелітки.

До Гуцульщини відносяться гірські райони Івано-Франківської і Чернівецької областей та Рахівський район Закарпатської області. Для кожного району, навіть для кожного села характерними є елементи індивідуальності, своє художнє лице. Кожне село прагне вдягнутися красиво і неповторно. Навіть по тому як вишита сорочка чи інший елемент одягу, можна дізнатися з якого села людина. Так для села Яворів улюбленими були мотиви «сливові», «черешневі», «соснові», «в очка», а для косівських майстрів - «скринькові» (поєднання ромбів, зигзагів, прямокутників). Для верховинської вишивки характерними є орнаменти утворені сполученням різноманітних ромбів і трикутників. Кожен орнамент подрібнений на невеличкі квадрати, «що як перлини, заповнюють весь внутрішній простір». Вишивка села Космач виділяється жовтогарячим колоритом. В ній поєнуються різні тони жовтого, оранжевого, червоного кольрів з вкрапленнями зеленого та чорного. Улюблені мотиви - «дубовий лист», «сосонкові», «черешневі», «баранкові», «качурові», «павункові». Жителі Гуцульщини люблять прикрашати свій верхній одяг: кептарі, сердаки, кожухи. Вишивають вони його завжди щедро і барвисто. Тому й збережені в вишивках елементи індивідуальності, які є тісно пов’язані з колективними елементами «при строгому дотриманні традиційних законів вишивання» [43].

Локальні відмінності для жителів певних етнографічних районів приведено в табл. 1.1.

Таким чином, вишивки жителів різних областей відрізняються лише локальними відмінностями - техніками вишивання, мотивами чи улюбленими кольорами, а закони вишивання залишаються однакові.

Виділимо наступні типи мотивів:

- геометричні;
- рослинні;

- рослинно- геометричні;
- зооморфні.

Таблиця 1.1

Локальні відмінності вишивок етнографічних районів

Етнографічне районування території	Локальні особливості вишивок		
	Техніки виконання	Типові орнаментальні мотиви	Колірна гама
1	2	3	4
Середнє Подніпров'я	низь, хрестик, гладь	рослинні - барвінок, хмелик, гілки, «ламане дерево», чернігівська берізка, грона винограду, ягід; геометричні - скісний та прямий хрест, квадрат, ромб, трикутник, зірки	голубий, зелений, сірий, пастельний, білий
Полісся	заволікання, гладь, занизування, застелювання	рослинний - шашечки, ключики, терен, виноград, хміль; геометричний -.ромб, ламані лінії, 8-микутні зірки, розетки; зооморфний - птахи	червоний, чорний
Поділля	низь, настил, хрестик, мережка, гладь	геометричний - розетка, меандрові завитки, ромби з виступами, гачки; рослинний -"солов'їні вічка", "зерновий вивід", "довбанка", калина, вазони	червоний і чорний, житній

Продовження табл. 1.1

1	2	3	4

Буковина	бісер, виколювання, стебнівка,	зооморфний - багатоногі птахи, зозульки, голуби, павуки;	багатоколірність в вишивці
	гладь, мережка	рослинний - дерева, галузки, квіти, колоски, геометричний - ромб, розетка, людські постаті;	
Карпати і Прикарпаття	хрестик, стебнівка, бісер, низь, стеклярус, металічні лелітки	геометричний - ромби, зигзаги, прямокутники багатопроменева розетка, «скринькові», меандр. рослинний -" квітки, галузки, «сливові», "черешневі", "соснові", "в очка"	червоний, синій, жовтий, чорний, зелений

Найпоширенішими геометричними мотивами є: ромб; трикутники; розетка; хрестоподібні; зірковидні фігури та ін.

Найпоширенішими рослинними мотивами є: квіти (троянди, лілії,); листки; пуп'янки; плоди (виноград, черешня), дерева (дуб, калина) та ін.

Серед зооморфних мотивів найчастіше зустрічаються : птахи, тварини і ін.

За критерій поділу вишивальних орнаментів візьмемо їх структурну композицію (підрозділ 2.1).

Мистецтвознавці весь узор зображений на рис. 1.1 називають орнаментом. Ми будемо надалі притримуватися цієї термінології. Орнамент складається з підорнаментів. Введемо поняття підорнамента. Підорнаментом будемо називати узор, що складається з ритмічно впорядкованих однакових елементів (побудований на одній групі перетворення). Підорнаменти діляться на рапорти [37]. Рапортом називається мінімальна по площі область якою можна покрити підорнамент використовуючи лише переноси. Рапорт в свою чергу ділиться ще на менші частинки - мотив або в нашій термінології мінімальний рисунок - $D\min$.

Мінімальний рисунок - це найменша частина рапорту, з допомогою якої, проводячи перетворення симетрії, будуємо рапорт.

Рис. 1.1. Зразок орнаменту і його складових.

Орнамент (рис. 1.1) складається з двох підорнаментів (позначення - I і II). Рапорт I-ого підорнаменту 1, мінімальний рисунок (обведено штриховою лінією) - 2; рапорт II-го підорнаменту - 3, мінімальний рисунок - 4. Отже, для орнаменту і його складових буде дійсною запропонована схема - рис. 1.2, де N - кількість підорнаментів в орнаменті, K, ..., L - кількість рапортів в підорнаментах, m, o, p, q - кількість мінімальних рисунків в рапорті.

Виділимо наступні характеристики орнаменту:

- коефіцієнт заповнення $k_з = \frac{S_{орн}}{S}$; $S_{орн}$ - площа, яку займає орнамент, S - загальна площа, виділена для синтезу орнаменту;
- кількість підорнаментів в орнаменті - $m_{по}$;
- кількість мінімальних рисунків в рапорті - $m_{мр}$;
- кількість рапортів в групі - $m_{рп}$.

В теорії і практиці моделювань і побудови розпізнаючих систем досліджують зображення різного походження. У зв’язку з цим розглядають прості і складні зображення. До простих двовимірних зображень часто відносять класи випуклих зображень. Складні зображення як правило це зображення довільної геометричної форми. У ряді випадків під складними зображеннями розуміють зображення довільної геометричної форми з можливими вирізами всередині також довільної геометричної форми. Власне за означенням симетричні зображення - орнаменти відносяться до класу складних зображень довільної геометричної форми і вирізами також довільної геометричної форми. До простих зображень-орнаментів ми будемо

відносити мінімальний рисунок D_{min} зображення-орнамента, тобто складову частину зображення (яка не має симетричної структури).

Таким чином, за означеннями, клас симетричних зображень-орнаментів є дуже широким класом складних зображень. Він охоплює широкий клас можливих складних зображень, які досліджуються в прикладних технічних розробках різного призначення.

Симетричне зображення-орнамент синтезується з простих зображень у вигляді мінімальних рисунків D_{min}, при цьому кількість мінімальних рівна 1.

Таким чином, на основі поняття симетрії розглянено українську народну вишивку, як об'єкт, наділений структурою, виділено структурні компоненти орнаменту та його характеристики. Крім цього, проаналізовані вишивки всіх етнографічних районів України, виділені їх локальні відмінності та проведено класифікацію вишивки по мотивах.

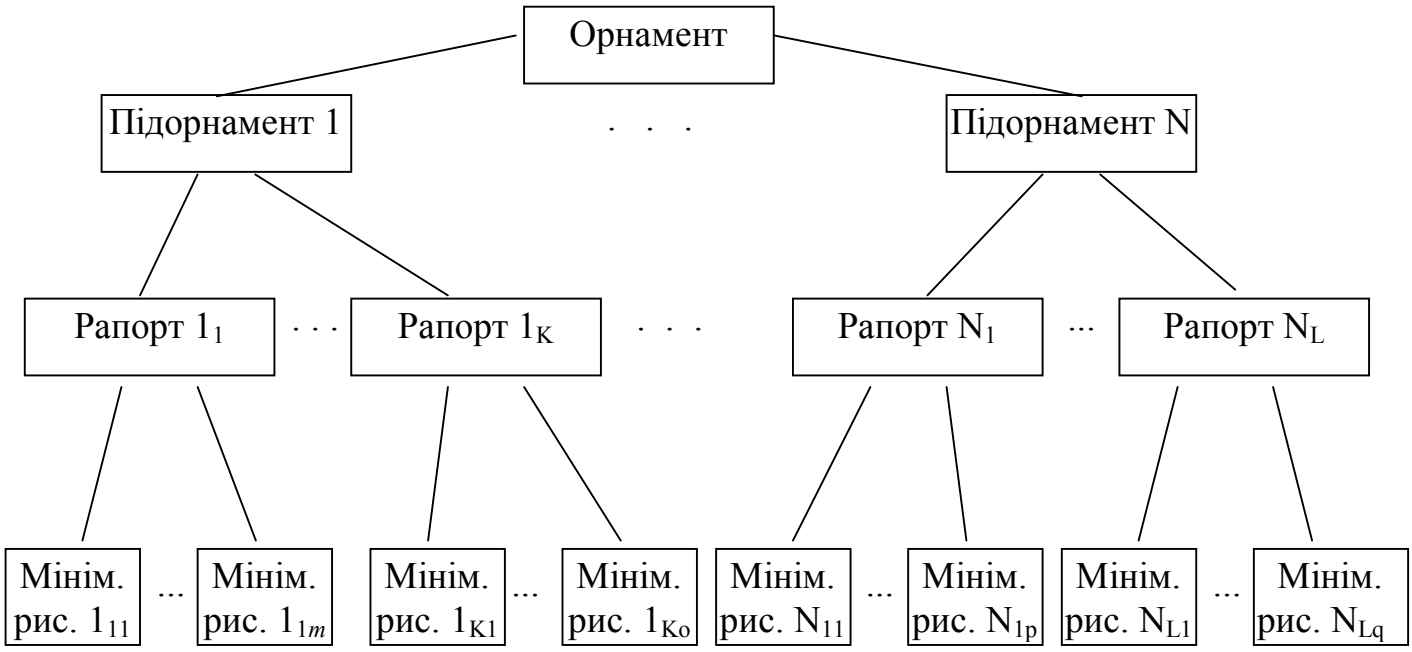


Рис. 1.2. Структура орнаменту.

1.3. Аналіз методів та програмних засобів комп'ютерної графіки

1.3.1. Комп'ютерна графіка - актуальний напрямок сучасних інформаційних технологій

Комп'ютерна графіка - це сукупність методів та засобів автоматизації процесів підготовки, вводу, перетворення, зберігання та відображення графічної інформації за допомогою обчислювальних засобів і графічних пристроїв. Як відомо [15-18] комп'ютерна графіка складається з двох основних частин:

1) технічне забезпечення - ЕОМ та периферійні пристрої;
2) програмно-математичне забезпечення (методи та алгоритми, машинні мови, елементи операційних систем, пакети прикладних програм.

На сучасному етапі виділяються наступні напрямки розвитку комп'ютерної графіки:

1) ілюстративна та ділова графіка (наукові дослідження, системи управління, виробництво);
2) автоматизована обробка зображень (зберігання, передача, аналіз та розпізнавання образів та зображень);
3) спеціалізована графіка (поліграфія, реклама);
4) графіка для використання в автоматизованих системах проектування;
5) комп'ютерна мультиплікація;
6) навчальна та ігрова графіка.

По типу подання графічної інформації комп'ютерна графіка поділяється на векторну та растрову. Векторна графіка подає зображення шляхом команд та математичних виразів. При використанні даного типу графіки кожен графічний об'єкт має своє представлення, тому дану графіку називають об'єктно-орієнтованою. Переваги даного типу графіки наступні:

- компактне зберігання графічної інформації;
- можливість збільшення векторного рисунку без втрати його якості;
- можливість редагування частин зображення без впливу на решту зображення.

Недоліком векторної графіки є затруднення при виведенні графічних об'єктів

на друкуючі пристрої.

При растровому типу графіки зображення подається у вигляді растру (двовимірного масиву точок), які упорядковані у рядки та стовці. Для кожної точки незалежно задаються колір та яскравість. Перевагами растрової графіки є:

- більша ефективність, порівняно з векторною графікою, подання реальних об’єктів;
- зручність виводу на друкуючі пристрої графічних об’єктів.

Недоліком растрової графіки є те, що для збереження графічної інформації необхідно великий об’єм пам’яті, тобто ставляться великі вимоги до апаратної частини комп’ютера.

В основі побудови векторної та растрової графіки лежать відповідні методи та алгоритми синтезу, опису та архівування зображень, які розглянемо в наступному пункті.

Таким чином, комп’ютерна графіка, як нова інформаційна технологія, знайшла широке застосування в різних галузях народного господарства, тому розробка редактора зображень має наукове та технологічне значення.

1.3.2. Методи опису, синтезу та архівування складних зображень

Розглянемо методи опису, синтезу та архівування складних зображень. В роботі [51] в якості математичного апарату в моделях морфології рослин використовуються L-системи. Для опису моделі запропоновано спеціальний клас атрибутних L-систем, які породжують геометричні образи. Елементарні образи (листок, фрагмент стебла) формують за допомогою простого геометричного редактора. Потім задають правила породжуючих образів. За допомогою правил з елементарних образів формується геометрична модель рослини і моделюється процес росту. До геометричного образу можуть бути прикріплені один або декілька нових породжуючих символів. Породжуючий символ має координати і напрямок в просторі. На кожному наступному кроці для кожного породжуючого символу шукається потрібне правило. В відповідності з ним відбувається перетворення породжуючого символу в образ.

При цьому можуть появлятися нові породжуючі символи з новими значеннями атрибутів. Процес закінчується тоді, коли не залишилося жодного породжуючого символу. L-системи подібні до граматик, проте якщо в граматиці на кожному кроці виводу змінюється єдине входження нетерміналу, то в L-системі всі нетермінали замінюються паралельно. Ця властивість дозволяє моделювати розвиток об'єкту.

В роботі [52] розглянута G-граматика, яка породжує пейзажі в стилі монохромної середньовічного живопису (використано лінгвістичний спосіб побудови зображення). Автори статей [53-55] розглядають поняття "фрактал" на основі якого проводиться синтез зображень. Існують наступні способи побудови фракталів:

1) за допомогою L-систем (граматики деякої простої мови);
2) за допомогою системи ітеруючих функцій - сукупності стискуючих афінних перетворень і ін.

При генеруванні зображень використовується підхід, базований на теорії нейронних мереж. Автори застосували нейронні мережі з зворотнім звязком для генерування геометричних орнаментів [56, 57]. В інших роботах [58, 59] на основі використання логістичної карти, побудованої на базі рекурентних функцій, приведено приклади генерування зображень детермінованим та випадковим чином.

Синтаксичний аналіз дає деякий структурний опис речення (в вигляді дерева або графа відношень). За допомогою синтаксичного підходу до розпізнавання образів досягається можливість, використовуючи невелику кількість непохідних елементів і граматичних правил, описувати велику кількість складних об'єктів [11-13, 60-62]. "Одним із найбільш приваблюючих аспектів цієї можливості є використання рекурсивної природи граматик" [25].

Засоби опису образів в термінах непохідних елементів і їх відношень, називаються «мовою опису образів» [24]. Правила цієї мови, які визначають способи побудови образу з непохідних елементів (композиції елементів), часто задають з допомогою так званої "граматики мови опису образів".

Ідею про те, що в якості методу опису мови можна використовувати породжуючу граматику з достатньо великою множиною виводу формалізував і

розвинув Хомський, Бар-Хіллел і їх співробітники [24, 63]. Хомський розділив породжуючу граматику по формі правил підстановки на 4 типи.

Граматики типу 0.

В цій граматиці є відсутніми будь-які обмеження на вид правил підстановки, тобто будь-які ланцюжки можуть стояти як зліва так і справа від стрілки. Це дуже широкий клас для використання.

Граматики типу 1 (граматики безпосередньо складових).

Правила підстановки цих граматик мають наступний вигляд:

$$\xi_1 A \xi_2 \rightarrow \xi_1 \delta \xi_2,$$

де $A \in V_N$, ξ_1, ξ_2, $\delta \in V^*$. Це правило читається як "А можна замінити на δ в контексті ξ_1, ξ_2".

Граматики типу 2 (безконтекстні граматики).

Правила підстановки цих граматик мають наступний вигляд:

$$A \rightarrow \delta,$$

де $A \in V_N$, $\delta \in V^+$. Допоміжний символ А згідно таких правил, замінюється ланцюжком δ незалежно від контексту , в якому зустрічається А.

Граматики типу 3 (автоматні або регулярні)

Правила підстановки цих граматик мають вигляд:

$$A \rightarrow aB \text{ або } A \rightarrow в,$$

де, $A, B \in V_N$, $a, в \in V_T$. А, В , a, $в$ - одиночні символи V.

Структурні методи розпізнавання базуються на породжуючій граматиці - системі, що складається з четвірки [24]:

$$G=(V_N, V_T, P, S),$$

де V_N, V_T - основний і допоміжний словники.

V_T - це набір вихідних елементів, з яких будують ланцюжки, які породжуються граматикою.

V_N - це набір символів, якими позначаються класи вихідних елементів або ланцюжків вихідних елементів, а також в окремих випадках деякі спеціальні елементи.

Р - скінченна множина правил підстановки, яка позначається "$x \rightarrow y$ ", що

означає "x замінити на y".

S - початковий символ ($S \in V_N$), який позначає сукупність всіх тих мовних об'єктів, для опису яких призначається дана граматика.

На початку 80-х років появилися методи представлення відеоданих на основі пірамідальних структур. Назва ця походить від слова піраміда, тому що зображення в інформаційному полі послідовно упорядковані і йдуть один за другим. Зображення при такому представленні розбиваються на рівні блоки, останні в свою чергу знову на блоки і до тих пір, поки блок не буде співпадати з елементом вихідного зображення. В результаті одержується набір зображень, кожне з яких складається з блоків відповідної величини і кожне з яких послідовно уточнює один одного і всі збігаються до вихідного.

Структуру взаємозв’зків зображень із піраміди найкраще описує рекурсія адже для побудови структури досить задати елементарне зображення і закон переходу на наступний рівень. Термін «рекурсія» походить з латинського слова «re curcio», що в буквальному перекладі означає - повернення назад. Термін «рекурсивний» зустрічається в літературі стосовно різних об’єктів і процесів. Рекурсивні функції, числа, послідовності зустрічаються в конструктивній математиці і теорії алгоритмів [64], рекурсивні програми, рекурсивні ЕОМ - в обчислювальній техніці і програмуванні.

Проаналізуємо відомі методи стиснення графічної інформації. Для її стиснення (з метою зменшення роміру пам’яті) використовують два підходи [65-67]: зовнішнє стиснення (стиснення всього файлу за допомогою форматів ZIP, ARC) та внутрішнє стиснення, яке реалізоване всередині файлу за допомогою методів RLE, LZV, JPEG. Зовнішні формати стиснення дуже поширені, тому не вимагають детального пояснення. Розлянемо внутрішні методи стиснення.

Метод RLE - метод групового кодування, який проводить запис растрового файлу за допомогою пар - кількість однакових пікселів та їх значення, що йдуть поряд у файлі. Даний метод використовується у графічному форматі PCX, BMP. Він ефективний для графічних об’єктів, які мають велику кількість однакових кольорів. Метод неефективний для фотореалістичних зображень.

Метод LZV базується на пошуку всередині растрового зображення однакових областей та побудові таблиці кодів, які відповідають однаковим областям. Метод застосовується в форматах GIF, TIFF. Даний метод аналогічно попередньому неефективний для стискування фотореалістичних зображень.

Метод JPEG - метод кодування нерухомих зображень [68,69]. Величина зображення, яке може оброблятися згідно цього методу від 1-1 до 65535-65535 активних елементів і кожний активний елемент може вміщувати від 1 до 255 кольорових компонентів. При цьому піксел зображення представляється з точністю від 2 до 12 біт в режимі з втратами в вихідному зображенні і від 2 до 16 при використанні без втрат. Вихідне зображення ділиться на блоки 8-пікселів. Виділений блок перетворюється шляхом дискретно-косинусного перетворення (ДКП) аналогічно перетворенню Фур'є. Поскільки перетворення двохкоординатне то базисом ДКП - функції із збільшеними частотами в горизонтальні та вертикальні площинах. В JPEG включені два методи:

1) метод прогресивно-ієрархічного відновлення - зображення поступово відновлюється на приймальній стороні;
2) метод спектральної селекції - полягає в групуванні обчислень ДКП коефіцієнтів.

Останнім часом знайшли розвиток нетрадиційні технології комп'ютерної графіки [70]. Згідно цих підходів зображення розбивається на процедурні файли та файли-посередники. Таким чином економиться пам'ять та дисковий простір і час редагування зображень. Всі деталі зображень розбиваються на об'єкти. Інструкції описують взаємний зв'язок об'єктів. Перевага цього підходу - економія пам'яті. Недолік - при генерації зображення витрачається додатковий час.

Цей підхід знайшов відображення в наступних програмних пакетах:

1) LIVE PICTURE 2.5.1 - реалізує нову технологію комп'ютерної графіки FITS (FUNCTIONAL INTERPOLATING TRANSFORMATION SYSTEM) - система трансформації зображень методом функціональної інтерполяції) і базується на використанні процедурних файлів і файлів -посередників.
2) MACROMEDIA XRES 2.0 - реалізована технологія файлів-посередників.

Для обробки зображень використовуються нейронні мережі [57]. При цьому

зображення розбивається на множину блоків певної розмірності. Кожному блоку ставиться у відповідність нейронна мережа.

Як показано в підрозділі 1.2, орнамент має наступну чітко виражену структуру: підорнамент, рапорт, мінімальний рисунок. Для опису мінімального рисунку запропоновано структурний (синтаксичний) підхід [60]. Він базується на аналогії між структурою образів і синтаксисом мов і використовується в тих випадках, коли описується образ з допомогою більш простих підобразів, а кожен підобраз знову описується ще більш простими підобразами. Крім цього, для опису підорнаменту запропоновано рекурсивний підхід до груп перетворень зображень [31]. Стиснення складних зображень реалізовано шляхом їх математичного опису (у вигляді формул зображення складових орнаменту).

Отже, в даному пункті на основі проведеного аналізу відомих методів опису, синтезу та архівування складних зображень, обгрунтовано застосування структурного та рекурсивних підходів до опису та синтезу складових орнаментів, що дало можливість розробити метод опису та моделювання складних симетричних зображень.

1.3.3. Аналіз сучасних програмних засобів комп'ютерної графіки

Всю сукупність програмних засобів можна розділити на наступні складові:

- графічні компоненти операційних систем;
- графічні можливості мов програмування;
- графічні бібліотеки і пакети прикладних програм (ППП);
- графічні редактори.

Графічні можливості операційних систем та мов програмування розглядати не будемо через їх складний та обмежений спосіб доступу до ресурсів комп'ютерної графіки. Графічні бібліотеки та ППП мають набагато ширші можливості ніж операційні системи та мови програмування. Наприклад, бібліотека Basic Professional System 7.1 містить програми створення різноманітних графіків та гістограм ділової графіки. Бібліотека компілятора Borland C++ містить функції, які дозволяють

працювати з окремими точками, графічними примітивами, фрагментами зображення, шрифтами.

Серед ППП по фукціональним можливостям слід виділити пакет програм машинної графіки ГРАФОР [17]. У бібліотеці ГРАФОРа є більше 400 програм для реалізації побудови зображень трьохвимірних обєктів, графіків в різних системах координат, апроксимації, згладжування функцій сплайнами, рядами Фур'є, многочленами Чебишева і Без'є, побудови ліній рівня функцій двох змінних.

Графічні редактори дозволяють користувачеві використовувати готові інструментальні засоби для створення зображення, тобто вони є найбільш універсальними та простими в користуванні.

Лідером серед графічних редакторів векторної графіки є CorelDRAW (виробник фірма Corel) [26]. Це потужний пакет, що дозволяє створювати складні зображення за допомогою команд редагування та перетворення об'єктів. Синтез візерунків та створення та редагування програм сценаріїв здійснюється за допомогою векторних команд. Основні функціональні можливості графічного редактора CorelDRAW 9.0 наступні:

- побудова ліній, прямокутників, еліпсів, многокутників, спіралей;
- виділення об'єктів, маштабування та зміна геометрії об'єктів;
- імпорт та експорт файлів у форматах інших програм;
- робота з буфером обміну Windows 95.

Недоліком даного графічного редактора є те, що формування орнаменту відбувається детермінованим способом в ручному режимі, тобто відсутні функції автоматичного його генерування та моделювання.

Графічний редактор растрової графіки Adobe Photoshop забезпечує розробку та ретушування фотографічних зображень і створення на їх основі художніх зображень. Приведемо основні функціональні можливості графічного редактора Photoshop 3.0:

- можливість роботи з шарами зображень;
- виділення областей та робота з ними;
- підтримка моделей цифрових зображень та форматів графічних файлів;
- орієнтація під Windows 95.

Недоліками даного графічного пакету є:

- жорстка орієнтація на обробку готових зображень, а не на створення нових;
- зображення займають великий об’єм памяті.

Інтегроване представлення зображення, звуку та тексту знайшло відображення в сучасній технології - multimedia. Multimedia - сукупність технологій, які застосовуються для цифрового представлення, зберігання, відтворення і редагування аудіо і відеоінформації. Різновидністю мультимедійного програмного забезпечення є авторські системи. Авторська система – це програмне забезпечення, яке дозволяє створювати multimedia-програми без традиційного програмування. Прикладом є система AUthorware Professional для Windows [71]. В цій системі реалізовані піктограми анімації, взаємодії, обчислення, групування, відтворення, звуку, відео. Звук є одним із основних компонентів multimedia. В multimedia застосовуються два основних методи представлення звуку: MIDI і оцифрований звук. MIDI застосовується для синтезу звуку. Основна ідея в представленні складного звуку послідовністю коду. Оцифрований звук - послідовність чисел, які представляють собою амплітуди звукового сигналу, які беруться через відповідні проміжки часу.

Розглянемо переваги та недоліки сучасних програмних засобів по multimedia.

Asymetrix Multimedia Toolbook 4.0.

Переваги: прості засоби навігації, потужна мова OpenScript, потужні редактори multimedia і Script Editor.

Недоліки: редагування вимагає операції вирізання і вставки.

Macromedia Director 5.0.

Переваги: гнучкий інтерфейс, точна синхронізація, розробка крос-додатків, широкий вибір можливостей розповсюдження і відтворення.

Недоліки: вимагає володіння мовою сценаріїв Lingo, середні редактори multimedia.

Microsoft Visual Basic 4.0.

Переваги: потужні можливості програмування, доступ до баз даних і зв'язку з іншими додатками.

Недоліки: складність навчання, робота тільки в Windows.

Oracle Media Objects 1.0

Переваги: проста компоновка multimedia, розробка крос-додатків, високорівнева мова сценаріїв.

Недоліки: середні інструменти для анімації, редагування вимагає операції вирізу і вставки.

Strata MediaForg 2.0

Переваги: об'єктно-користувацький інтерфейс, спецефекти.

Недоліки: середня організація управління і синхронізації.

3D Studio Max-32 розрядний Windows NT додаток для трьохмірного моделювання, візуалізації і анімації.

Сучасною технологією по multimedia є MMD - Multi Media Data base. Він дозволяє поєднати multimedia технології з технологіями бази даних. Таке поєднання забезпечує накопичення інформації, даних, фотографій, музичних файлів, кліпів, текстів і дозволяє користуватися технологією бази даних. MMD має наступні функціональні характеристики:

- об'єктно-орієнтоване відношення;
- основа побудована на принципах баз даних;
- вмонтований виклик комплектного тексту;
- підтримка гіпертексту;
- розширена media-підтримка: 3 image-форматів, є vector-форматів, 6 відео-форматів і багато word-processing форматів;
- image підтримка - можливість зменшення, збільшення зображення;
- компресорна підтримка;
- інтегрована OSR - підтримка дозволяє накопичувати друковану інформацію, викликати її і бачити в оригінальному розташуванні сторінок;
- підтримка бази даних.

MMD підтримує всі стандартні, а також багато нестандартних multimedeia - data-формати.

На основі MMD технології розроблені наступні програмні пакети:

- MMD/Gen.+ підтримує multimedia-Database (сотні і тисячі зображень і включає

вмонтовані OCR;

- MMD/Gen.: для застосування з Database середньої величини і максимально до 200 зображень;

- MMD/учбовий, спрощена версія для застосування в школах, максимально до 50 зображень;

- MMD/RT для загальних застосувань MMD;

- MMD/RTK: пакет запасних частин для CD-Rom.

Отже, із проведеного аналізу графічних редакторів видно, що в них відсутні функції автоматичного генерування та моделювання зображень-орнаментів, тому актуальною є задача розробки спеціалізованого редактора орнаментів, який дозволить генерувати та ефективно зберігати складні симетричні зображення.

Таким чином, в даному пункті проаналізовані відомі програмні засоби комп'ютерної графіки: їх переваги та недоліки, доказана необхідність розробки редактора орнаментів.

1.4. Постановка задач дисертаційного дослідження

Із проаналізованих методів та програмних засобів комп'ютерної графіки випливає, що даний напрямок нових інформаційних технологій є актуальним і перспективним через широту свого практичного застосування. Проведений аналіз в підрозділах 1.1-1.3, показує, що проблемами синтезу та моделювання складних зображень-орнаментів на Україні не займалися, тобто немає комп'ютерно-технологічних розробок і моделей опису та синтезу симетричних зображень. Розробка математичних моделей опису, синтезу і моделювання зображень-орнаментів (вишивок) та створення на їх основі редакторів орнаментів та вишивок має не тільки наукову, але й практичну і технологічну цінність. Дані редактори є ядром нової інформаційної технології побудови, зберігання та виготовлення складних зображень в текстильній, легкій промисловостях, поліграфії тощо.

Отже, метою дисертаційного дослідження є розробка методу опису та моделювання складних симетричних зображень-орнаментів та створення на його

основі редактора зображень-орнаментів вишивки. Структура мети і задач дослідження приведені на рис.1.3. Для вирішення мети теоретичного дослідження необхідно розв’зати наступні задачі:

* розробити метод опису та моделювання складних симетричних зображень-орнаментів;

* розробити алгоритми синтезу та моделювання складних симетричних зображень-орнаментів.

Розв’язання мети експериментальних досліджень передбачає розв’язання таких задач:

- програмну реалізацію алгоритмів синтезу та моделювання орнаментів (редактор орнаментів);
- програмну реалізацію алгоритмів редагування вишивок (редактор вишивок).

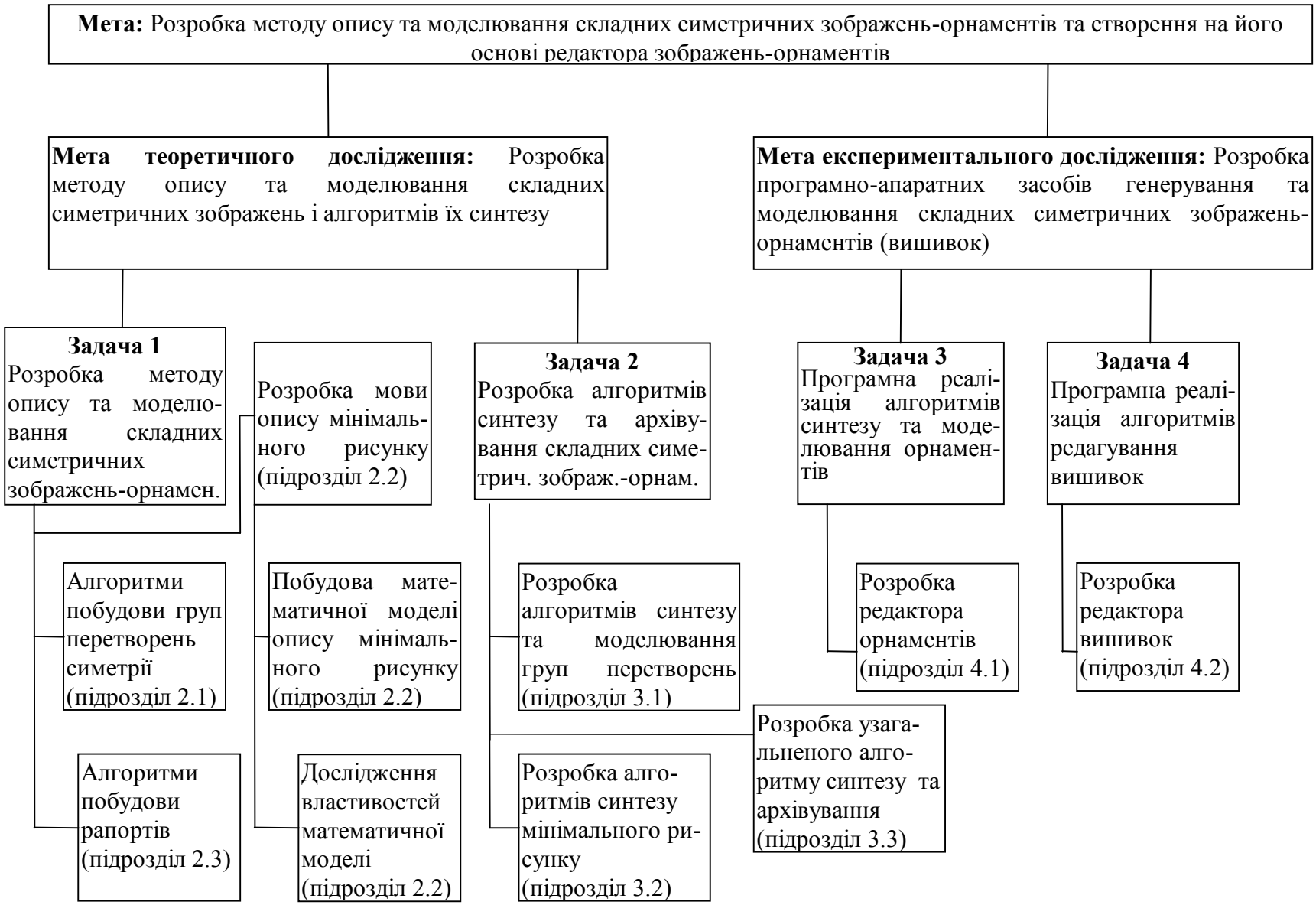


Рис. 1.3. Структура мети і задач дослідження.

1.5. Основні результати, отримані у розділі 1

1. Показано фундаментальну роль поняття симетрії (в науці, техніці, мистецтві, побуті), розкрито її геометричну суть та приведено групи перетворень симетричних зображень для площини та смуги.

2. На основі поняття симетрії розглянуто українську народну вишивку, як об’єкт, наділений симетричною структурою, виділено структурні компоненти орнаменту (вишивки) та його характеристики. Крім цього, проаналізовані вишивки всіх етнографічних районів України, виділені їх локальні відмінності та проведено класифікацію вишивки по мотивах.

3. Проаналізовані основні методи опису, синтезу, моделювання та архівування складних зображень, розглянуті програмні засоби їх реалізації, обгрунтовано застосування структурних методів при описі зображень-орнаментів (вишивки), сформульована мета дисертаційного дослідження та основні задачі для її реалізації.

РОЗДІЛ 2
РОЗРОБКА МЕТОДУ ОПИСУ ТА МОДЕЛЮВАННЯ СКЛАДНИХ ЗОБРАЖЕНЬ-ОРНАМЕНТІВ

В підрозділі 1.2 показано, що вишивка складається з ритмічно впорядкованих однакових елементів, тобто володіє певною симетричною структурою. В її основі лежать наступні структурні складові: орнамент, підорнамент, рапорт та мінімальний рисунок (див. рис.1.2). В підрозділі 1.3 обгрунтовано застосування структурного підходу до аналізу та моделювання складних симетричних зображень. Отже, завданням даного розділу є розробка методу опису та моделювання складних симетричних зображень. Як відомо [72], метод в обчислювальній математиці - це множина алгоритмів, які пов’язані загальною ідеєю побудови. Даний метод передбачає розробку наступної множини алгоритмів: перетворення над зображеннями вишивки (формалізація груп перетворень); мови опису зображень (формалізація мінімального рисунку); формалізацію рапортів (рис.2.1).

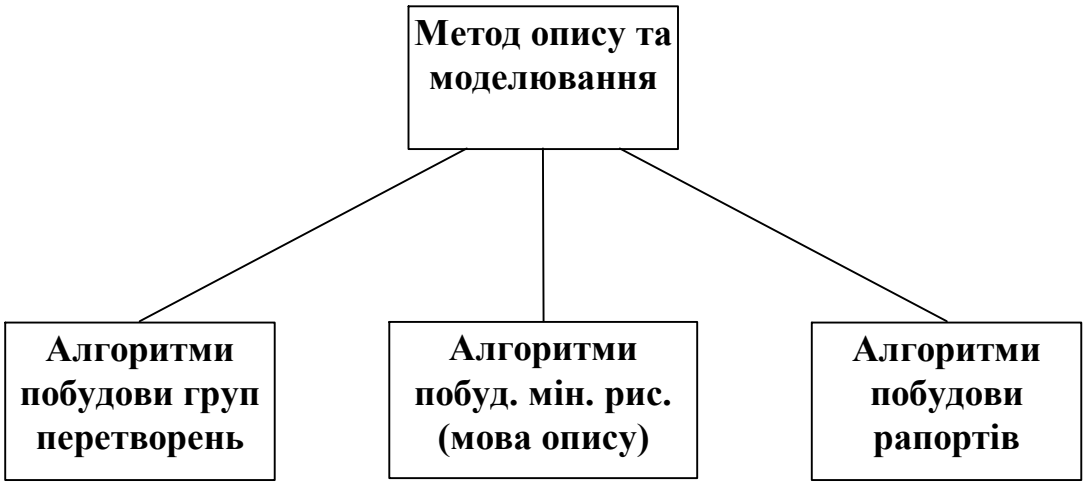


Рис. 2.1. Структура методу опису та моделювання складних зображень-орнаментів.

2.1. Перетворення над зображеннями-орнаментами (вишивки)

Розглянемо зображення-орнамент (вишивку) з математичної точки зору, тобто як об'єкт, наділений структурою. Згідно теорії Евкліда структуру простору можна описати з допомогою ряду основних співвідношень між точками [2, 73], - такими як:

точки A,B,C лежать на прямій, точки A,B,C,D лежать на площині, відрізок AB конгруентний відрізку CD.

Нехай дано відображення S:p→p'. Два відображення S, S': p→p', p'→p називаються взаємнооднозначними відображеннями або перетвореннями. Відображення від площини - основна операція дзеркальної симетрії - володіє тією властивістю, що двократне виконання цієї операції SS' - приводить до тотожнього відображення. Якщо відображення S - перетворення, оберненим для якого є S^{-1}, то S^{-1}- також є перетворення, і оберненим для нього є S. Добуток 2-х перетворень ST є знову перетворення і $(ST)^{-1} = T^{-1}S^{-1}$. По Лейбніцу [2], перетворення, що залишають незмінною структуру простору, називаються автоморфізмами. Тотожнє відображення є автоморфізм і якщо S - автоморфізм, то S^{-1} також автоморфізм. Крім цього, ST - знову таки автоморфізм (ST знову перетворення). Все це іншим способом виражено твердження, що

1) будь-яка фігура подібна сама собі;
2) якщо F' подібна F, то F подібна F';
3) якщо F подібна F', F' подібна F'', то F подібна F''.

Це - означення математичного терміну групи. Отже, автоморфізми утворюють групу. Всяка сукупність, всяка множина перетворень Г утворює групу, якщо виконуються наступні умови:

1) тотожнє перетворення I належить множині Г;
2) якщо перетворення S належить множині Г, то і S^{-1} також належить множині Г;
3) якщо перетворення S і T належать Г, то і ST належить Г.

Один із способів опису структури простору [2] полягає в використанні поняття конгруентності. Конгруентні частини простору V і V' - це такі його частини, які можна заповнити одним і тим же твердим тілом в 2-х його положеннях. Якщо тіло пересувається з одного положення в інше, то частинка тіла, що займала точку p в V, займе положення деякої точки p' в V'. Результатом руху є відображення p→p' частинки простору V на частинку простору V'. Межі твердого тіла можна розширити, так щоб воно охоплювало довільно задану точку p простору; таким

чином конгруентне перетворення є подібністю або автоморфізмом.

Конгруентні перетворення утворюють групу, яка є підгрупою групи автоморфізмів. Рухи - це такі перетворення подібності, які не змінюють розмірів тіла. В свою чергу рухи поділяються на власні і дзеркальні. Власні рухи, згідно Вейля, переводять лівий "гвинт" в лівий і правий в правий. Дзеркальні рухи переводять лівий на правий і навпаки. Рухи утворюють підгрупи групи подібностей або автоморфізмів. Власні рухи є групою, а дзеркальні рухи групою не є, так як добуток 2-х дзеркальних рухів є власний рух.

До руху відноситься і поворот. При повороті вся площина повертається навколо деякої точки на заданий кут. В результаті розміри і форма будь-якої фігури залишаються незмінними. Центр повороту (який може належати або не належати фігурі, що обертається) є єдиною нерухомою точкою. Повороти є власні і дзеркальні. Повороти навколо даного центру утворюють групу.

Найпростішим видом рухів є переноси. Перенос зручно представляти вектором. Послідовне виконання двох переносів AB і BC дає в результаті перенос AC. Отже, переноси утворюють групу.

Нижче проведемо дослідження симетрії фігури. Якщо задана деяка фігура, то всі перетворення, що не змінюють її, утворюють групу, яка дає точний опис симетрії фігури. Розглянемо, наприклад, рухи вишивального геометричного мотиву зображеного на рис. 2.2.

За елементи групи, що нас цікавить, будемо розглядати рухи в яких переміщений мотив буде суміщатися з вихідним. Для зручності позначимо вершини фігури цифрами і одну з вершин кружечком. Для суміщення даної конфігурації не обов'язково щоб кожна з позначених вершин співпала з собою, треба тільки, щоб множина точок, які складають вітки фігури після повороту, співпала з множиною точок в початковому положенні. Повернемо мотив на кут $p/2$ навколо точки O проти годинникової стрілки. Повернута фігура є другою фігурою, накладеною на фігуру, що знаходиться в початковому положенні (рис.2.3). При такому повороті відбувається перестановка вершин, а саме: вершина 1 заміщається

вершиною 2, вершина 2 - вершиною 3, вершина 3 - вершиною 4, вершина 4 - вершиною 1.

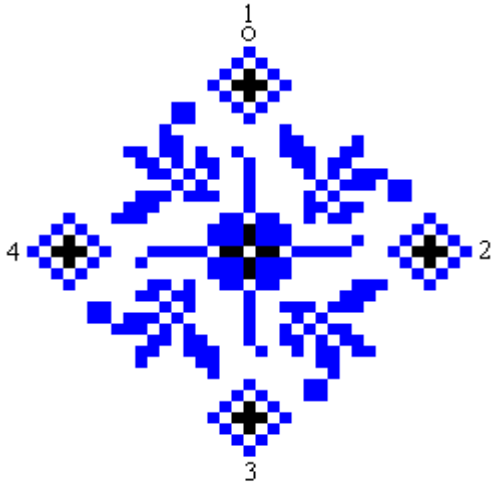
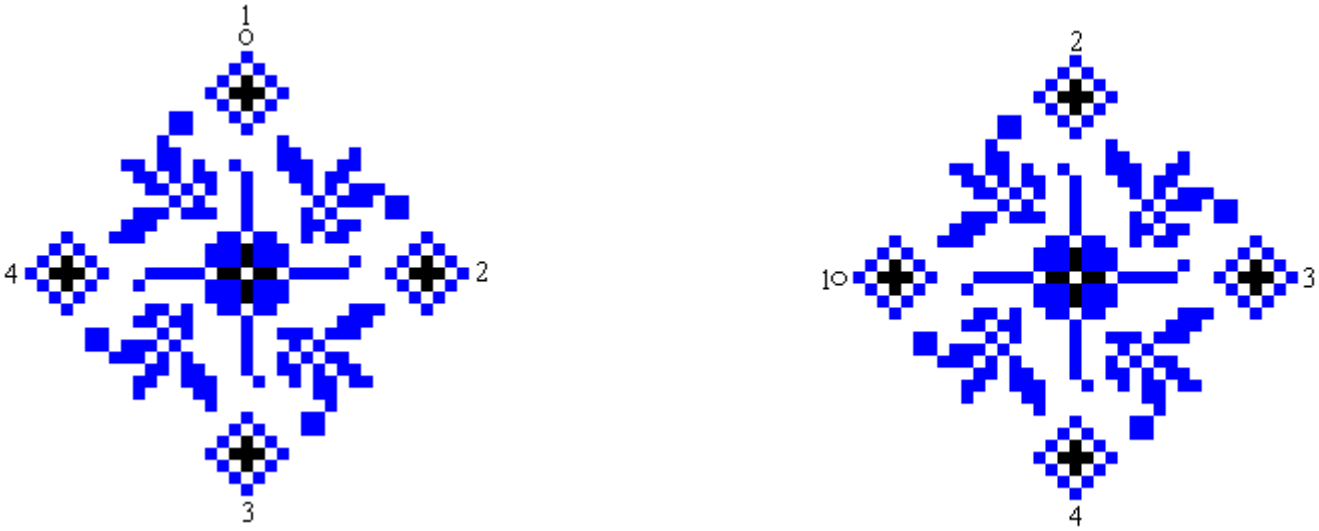


Рис. 2.2. Геометричний мотив.



а) б)

Рис. 2.3. Вихідне положення (а), після повороту на кут $\pi/2$ (б).

Всі повороти, які переводять мотив з вихідного положення в друге положення, зображене на рис. 2.3(б) можна записати у вигляді нескінченної множини:

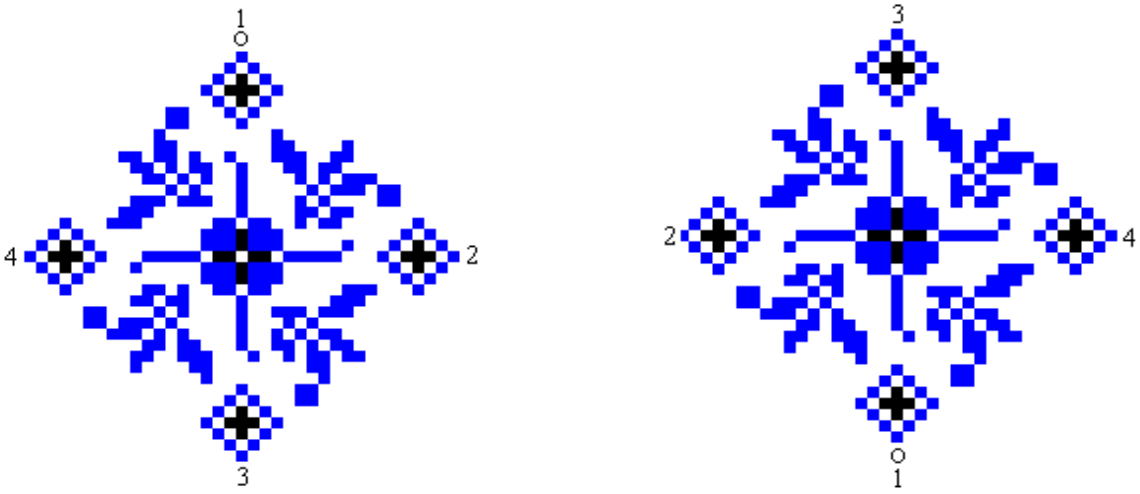
A={повороти проти годинникової стрілки на $\pi/2 \pm 2\pi k$, k=0,1,2,...}.

Повороти проти годинникової стрілки на від'ємний кут - це є повороти за годинниковою стрілкою. Всі рухи з множини А мають однакову властивість, адже вони переводять вершини фігури таким чином:

Поч. розміщення	Розміщення після повороту
1	2
2	3
3	4
4	1

Причому ця властивість зберігається незалежно від вибору початкового розміщення для мотиву. Позначимо через a деякий елемент множини A, а саме поворот на $\pi/2$ (k=0). Вершини в цій ситуації об'єднуються в пари наступним чином: a: 1→2, 2→3, 3→4, 4→1.

Розглянемо множину поворотів B={повороти проти годинникової стрілки на $\pi \pm 2\pi k$, k=0,1,2,...}. В результаті кожного з цих поворотів проходить накладання, яке показано на рис.2.4.



а) б)

Рис. 2.4. Вихідне положення (а), після повороту на кут π (б).

Повороти множини B не входили в множину A. Множина B теж переводить фігуру саму в себе. Аналогічно, як в попередньому випадку, виберемо довільний елемент множини B. Нехай b- поворот на π (k=0). b переводить вершини так: b: 1→3, 2→4, 3→1, 4→2.

Розглянемо множину поворотів C={повороти проти годинникової стрілки на $2\pi/3 \pm 2\pi k$, k=0,1,2,...}, яка теж переводить розглядувану фігуру в себе. В результаті кожного з цих поворотів проходить накладання, яке призведе до

наступних перетворень. Нехай *c*- поворот на 2p/3 (k=0): *c*: 1→4, 2→1, 3→2, 4→3.

Є ще одна множина рухів, які переводять дану конфігурацію в себе, - це множина *D*={повороти проти годинникової стрілки на $2\pi \pm 2\pi k$, k=0,1,2,...}. Довільний елемент *d* множини *D* - це накладання: 1→1, 2→2, 3→3, 4→4. Будь-який елемент множини D - це одиничний елемент групи поворотів фігури, тому що, *ad=da=a*, *bd=db=b*, *cd=dc=c*, *cc=c*, і результат не залежить від того, які елементи *a*, *b*, *c*, *d* вибрані з множин *A*, *B*, *C*, *D* відповідно. Для довільного елемента групи *D* введемо позначення *I*.

Позначимо через *Г*={*A*,*B*,*C*,*D*} і покажемо, що *Г* є група. Для цього знайдемо добуток (суперпозицію поворотів) будь-яких двох елементів.

Внаслідок суперпозиції рухів *a* і *b* одержимо:

ab: 1→2→4 або 1→4

2→3→1 або 2→1

3→4→2 або 3→2

4→1→3 або 4→3

Отже, *ab=c*.

Аналогічно знаходимо *bc*:

bc: 1→3→2 або 1→2

2→4→3 або 2→3

3→1→4 або 3→4

4→2→1 або 4→1

Отже, *bc=a*.

Легко можна переконатися, що *aa=b*, *ac=I*, *ba=c*, *bb=I*, *ca=I*, *cb=a*, *cc=b*.

Розгляд всіх можливих добутків показав, що операція суперпозиції поворотів не виводить за межі множини *Г*. Перевіримо також і групові аксіоми.

Асоціативність: (*ac*)*b* =*a*(*cb*), *cc=aa=b* і т.д. аналогічно можна довести для інших елементів.

Одиничний елемент. Існування одиниці доведено при розгляді множини D.

Обернений елемент. Так як *ac=I*, *bb=I*, *ca=I*, *II=I* то кожен елемент в нашій

множині володіє оберненим.

Побудуємо таблицю множення для групи поворотів *Г*. В верхньому її рядку розміщуються всі елементи групи, в цьому ж порядку розташовуються елементи зліва (табл. 2.1).

Таблиця 2.1

Таблиця множення групи поворотів *Г*

Множники	*I*	*a*	*b*	*c*
I	*II*	*Ia*	*Ib*	*Ic*
a	*aI*	*aa*	*ab*	*ac*
b	*bI*	*ba*	*bb*	*bc*
c	*cI*	*ca*	*cb*	*cc*

Використовуючи попередні результати, таблиця множення перепишеться у вигляді табл. 2.2:

Таблиця 2.2

Таблиця множення групи поворотів *Г*

Множники	*I*	*a*	*b*	*c*
I	*I*	*a*	*b*	*c*
a	*a*	*b*	*c*	*I*
b	*b*	*c*	*I*	*a*
c	*c*	*I*	*a*	*b*

З табл. 2.2 видно, що всі елементи групи попарно перестановочні (комутативні) і що вона є симетричною відносно головної діагоналі. Оскільки, по означенню, група називається комутативною, якщо будь-які два елементи її комутативні [74], то група Г є комутативною. Якщо позначити добутки $aa=a^2$, $aaa=a^3$, $aaaa=a^4$ і використати одержані раніше результати, то отримаємо, що

$b=aa=a^2$, $c=aaa=a^3$, $I=aaaa=a^4=I$. Тоді табл. 2.2 перепишеться у вигляді табл. 2.3.

Таблиця 2.3

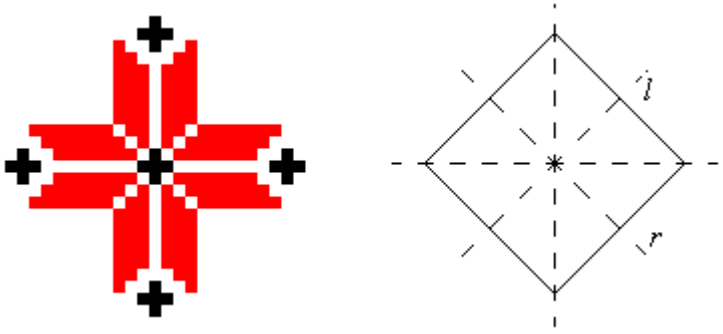
Таблиця множення групи поворотів Γ

Множники	I	a	a^2	a^3
I	I	a	a^2	a^3
a	a	a^2	a^3	I
b	a^2	a^3	I	a
c	a^3	I	a	a^2

Елемент a називається твірним (або твірним елементом групи) Г [74]. Якщо записати степені твірної a групи Γ: a, a^2, a^3, a^4, a^5, a^6, a^7, a^8, a^9,... і врахувати, що $a^4=I$, то одержимо послідовність a, a^2, a^3, I, a, a^2, a^3, I, a,..., в якій циклічно повторюється сукупність елементів a, a^2, a^3, I. Така група називається циклічною групою порядку 4 і позначається C_4.

Розглянемо тепер рухи вишивального геометричного мотиву, показаного на рис. 2.5.а. Фігура буде переводитися в себе поворотом на p/2 і відображенням відносно прямих l, r (рис. 2.5.б).

Таблиця множення набуде вигляду табл. 2.4, з якої видно, що операція "послідовного виконання" не виводить за межі заданої множини операцій, аксіома про обернений елемент виконана (так як I зустрічається один раз в кожному стовпчику і рядку), група некомутативна. Дана група в літературі [2, 35, 65] відома під назвою діедральна група порядку 4 - D_4.



а) б)

Рис. 2.5. Геометричний мотив (а); схема перетворень симетрії (б).

Таблиця 2.4

Таблиця множення груп поворотів і відображень

Множники	I	a	a^2	a^3	l	la	r	ra
I	I	a	a^2	a^3	l	la	r	ra
a	a	a^2	a^3	I	ra	r	l	la
a^2	a^2	a^3	I	a	la	l	ra	r
a^3	a^3	I	a	a^2	r	ra	la	l
l	l	r	la	ra	I	a	a	a
la	la	ra	l	r	a	I	a	a
r	r	la	ra	l	a	a	I	a
ra	ra	l	r	la	a	a	a	I

Отже, у вишивальних мотивах спостерігаються обидва види повороту [75], які переводять фігуру в себе, які породжують наступні групи:

1) група, що складається з повторних застосувань повороту на кут $\alpha = 2p/n$, де n-ціле число (циклічна група C_n);

2) група цих поворотів взятих разом з відображенням відносно n осей, що утворюють між собою кут $\alpha/2$ (діедральна група D_n) [2, 35, 74].

На рис. 2.6 показано два мотиви. Перший з них належить до групи симетрії C_1 (повна відсутність симетрії). Ця група найчастіше зустрічається у вишивці рослинних та зооморфних мотивів. Другий відноситься до групи D_1 (тут наявна одна дзеркальна симетрія). Цей тип симетрії теж досить часто зустрічається у вишивках.

Рис. 2.6. Рослинні мотиви.

Серед груп вищих порядків найчастіше зустрічаються у вишивках групи з n парним порядком, а саме з n=4, 8. Приклади з n=4 наведено на рис. 2.1, 2.4. Група C_5

зустрічається в природі як група симетрій п'ятипелюсткових квітів.

Вище приведені групи належать до типу груп симетрій, при яких на площині існує точка, яка буде залишатися нерухомою при всіх перетвореннях симетрії [76].

Розглянемо вишивальний орнамент, зображений на рис. 2.7.а. Він відноситься до іншого типу узорів, відмінного від попередніх. В попередніх випадках здійснювалися повороти зображень навколо деякої точки. В даному вишивальному орнаменті здійснюється перенос по прямій лінії.

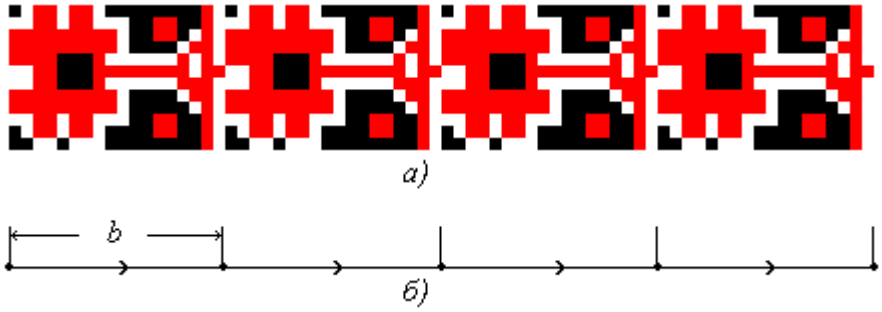


Рис. 2.7. Лінійний орнамент (а); схема перетворень (б).

Лінійні орнаменти досить добре вивчені вченими [2, 35, 77]. Якщо розглядати їх тільки як одномірні перетворення, то у випадку прямої лінії відображення може відбуватись від будь-якої точки О. Це відображення переводить деяку точку А прямої в А', яка лежить по другу сторону від О на тій же віддалі, що й А. Для одномірної прямої такі відображення є єдино можливими дзеркальними рухами, а переноси - єдино можливими власними рухами [2]. Аналогічно, як в попередніх випадках поворотної симетрії на площині можна виділити два типи орнаментної симетрії на прямій. Узор, який приведено на рис. 2.7.а відноситься до другого типу - симетрії, що не містить відображень (нескінченна циклічна група C_∞). Вишивка являє собою повторення елементів з правильним ритмом. Якщо b - це довжина елемента узору, що зображений на рис. 2.7.а, то перенос t зсуває узор на віддаль b, t - на віддаль 2b, t - на віддаль 3b. І в загальному випадку t пересуває елемент узору на віддаль nb, (де n=0, 1, 2,...). Схема такого перетворення показана на рис. 2.7.б.

Якщо ритміка переносу ще й супроводжується й дзеркальним відображенням, як на рис. 2.8.а, то такого роду вишивальний орнамент відноситься до першого типу перетворень. Аналогічно, якщо вважати, що довжина елемента узору дорівнює b, тоді центри відображень слідують один за одним на відстані b/2. Схема такого

перетворення наведена на рис.2.8.б. Група перетворень наведеної вишивки називається діедральною групою D_∞ [74].

Таким чином, в українській вишивці спостерігаються обидва типи груп симетрії лінійних орнаментів. Але часто, завдяки використанню різного кольору ниток, переносна симетрія спостерігається тільки в зовнішніх контурах вишивок, а внутрішня порушується.

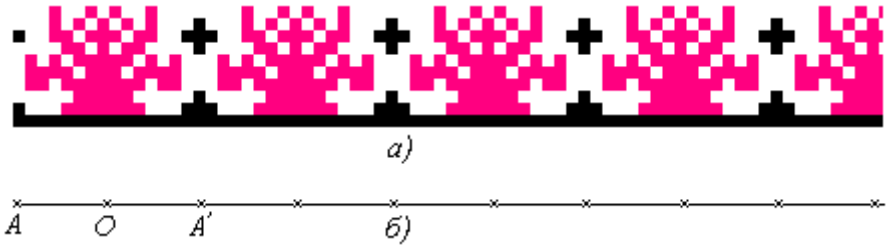


Рис. 2.8. Лінійний орнамент (а); схема перетворень (б).

Лінійні орнаменти української вишивки володіють не тільки одним виміром - довжиною, а й другим - шириною. Тому спостерігаються ще й інші типи симетрії. Це може бути симетрія відносно поздовжної прямої і симетрія відносно поздовжної прямої з переносом вздовж цієї прямої (ковзна симетрія). Існує сім груп орнаментів вишивки (підрозділ 1.2).

Вище приведені групи належать до типу груп симетрій, при яких на площині не існує точки, що буде залишатися нерухомою при всіх перетвореннях симетрії, але існує пряма, яка буде суміщатися з собою при названих перетвореннях [76]. На рис. 2.7.а, 2.8.а наведено узори української вишивки, які належать до груп перетворень p1, p1m відповідно. В додатку А приведені фрагменти вишивки всіх семи груп на смузі. Введемо позначення для множини груп на смузі J = { *p1, p2, pm, pg, cm, pmm, pmg, pm, pmm*}, j($D\min$) - зображення-орнамент на смузі, де j **є** J.

Часто у вишивках узор заповнює площину тканини не смугами, а частинками, які повторюються і по вертикалі і по горизонталі. Такого типу орнаменти відповідають деяким групам. Узори такого типу досить часто зустрічаються в архітектурних прикрасах, шпалерах, тканинах і т. д. Це - федоровські плоскі групи [25]. У всіх цих випадках власним рухом може бути або перенос, або поворот навколо точки О. Відомо [2], що кут повороту може приймати тільки значення π, $2/3\pi$, $\pi/2$, $1/3\pi$. Дзеркальним рухом є осьова симетрія, або ковзна. Для даних груп

не існує ні точки, ні прямої, які б суміщалися з собою при всіх перетвореннях групи (III-ій тип дискретних груп симетрій [76]). Однак в вишивках «хрестиком» (площина розділена на квадрати) не всі вони зустрічаються. Немає тих груп, для яких кут повороту може приймати значення $2/3p$, $1/3p$, виключаються відображення в сторонах рівностороннього трикутника, відображення в сторонах половини рівностороннього трикутника. Тому лише 12 груп є в вишивальних орнаментах* [32]. Приведемо їх позначення і породжувальні перетворення:

p1 - група, що одержується двома паралельними переносами;

p2 - група, що одержується трьома центральними симетріями;

pm - група, що одержується двома осьовими симетріями і паралельним переносом;

pg - група, що одержується двома ковзними симетріями з паралельними осями;

cm - група, що одержується осьовою і ковзною симетрією з паралельними осями;

pmm - група, що одержується симетрією відносно 4-ох сторін прямокутника;

pmg - група, що одержується однією осьовою і двома центральними симетріями;

pgg - група, що одержується двома ковзними симетріями з перпендикулярними осями;

cmm - група, що одержується двома осьовими симетріями з перпендикулярними осями і однією центральною;

p4 - група, що одержується центральною симетрією і поворотом на 90°;

p4m - група, що одержується симетрією відносно 3-х сторін прямокутного рівнобедреного трикутника;

p4g - група, що одержується осьовою симетрією і поворотом на 90°.

Позначимо множину груп перетворень на площині для вишивки $Z = \{ p1, p2, pm, pg, cm, pmm, pmg, pgg, cmm, p4, p4m, p4g\}$, $z(D_{\min})$ - зображення-орнамент на площині, де $z \in Z$.

Покажемо, що всі види симетрії можна отримати на основі відображення відносно осі (комбінацією осьових симетрій) (рис.2.9) [29].

*Примітка. В додатку Б приведені фрагменти вишивки 12-ти груп на площині.

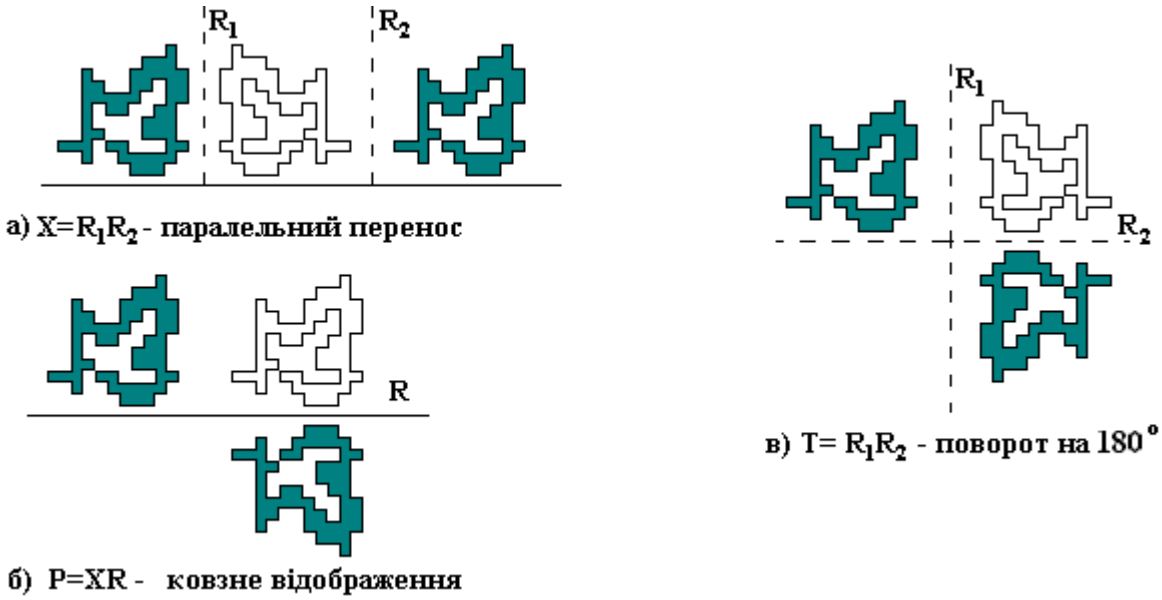


Рис. 2.9. Одержання перетворень симетрії на основі осьових симетрій.

Представимо множини груп Z і J через відображення [29, 30]. Нехай R_1, R_2, R_3 - відображення в гіпотенузі і катетах прямокутного рівнобедреного трикутника (рис. 2.10.а). Тоді множина груп Z буде представлятися таким чином:;

p4m	R_1, R_2, R_3;
p4g	R_2, $R_1 R_3$*;
p4	$R_1 R_3$, $R_2 R_3$;
cmm	R_1, $R_3 R_1 R_3$, $R_2 R_3$;
pgg	$R_2 R_1 R_3$, $R_1 R_3 R_2$;
cm	R_1, $R_2 R_1 R_3$;
pg	$R_2 R_1 R_3$, $R_1 R_2 R_1 R_3 R_1$;
pmg	$R_1 R_2 R_1 R_2 R_3$, $R_1 R_3 R_1 R_3$;
p2	$R_2 R_3$, $R_1 R_3 R_1 R_3$, $R_1 R_2 R_3 R_1$;
pmm	R_2, R_3, $R_1 R_2 R_1$, $R_1 R_3 R_1$;

*Примітка. Добуток осьових симетрій потрібно розуміти так $R_1R_3=R_1(D\text{min})R_3(R_1(D\text{min}))$, тобто проводимо відображення мінімального рисунка відносно R_1, потім утворений відображаємо відносно R_3. Результатом буде кінцеве зображення.

pm R_2 , $R_1 R_3 R_1$, $R_3 R_1 R_2 R_1$;

p1 $R_2 R_1 R_3 R_1$, $R_3 R_1 R_2 R_1$.

Аналогічно підійдемо і до множини груп J. Нехай R_1, R_2, R_3 - відображення в сторонах прямокутника (рис. 2.10.б). Множина груп J матиме наступне представлення:

p1 $R_1 R_3$;

pg $R_1 R_3 R_2$;

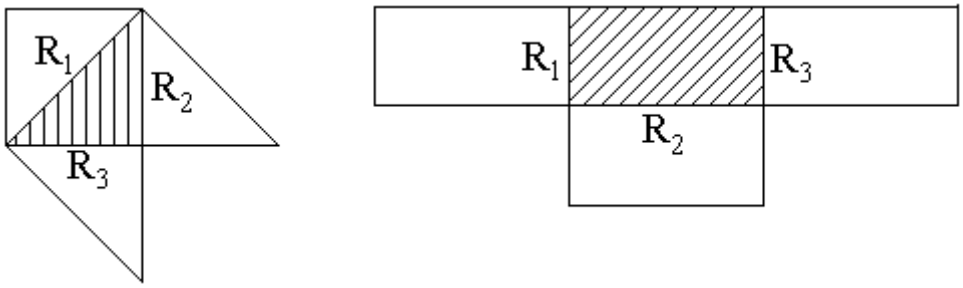
p1m R_1 , R_3;

p2 $R_1 R_2$, $R_3 R_2$;

pmg $R_1 R_2$, R_3;

pm $R_1 R_3$, R_2;

pmm R_1 , R_2 , R_3.



а) б)

Рис.2.10. Осі відображень для груп перетворень: для груп площини (а); для груп смуги (б).

На рис. 2.11 приведені приклади вишивок груп: а) p2 - для площини, б) p1 - для площини, в) pmm - для смуги відповідно. Всі ці зображення-орнаменти мають один і той же мінімальний рисунок.

Отже, в даному підрозділі доказано, що перетворення над зображеними вишивки утворюють групи, причому 12 груп перетворень на площині та 7 груп перетворень на смузі, які входять до відповідних федорівських груп, здійснено їх формалізацію на основі використання осьових симетрій.

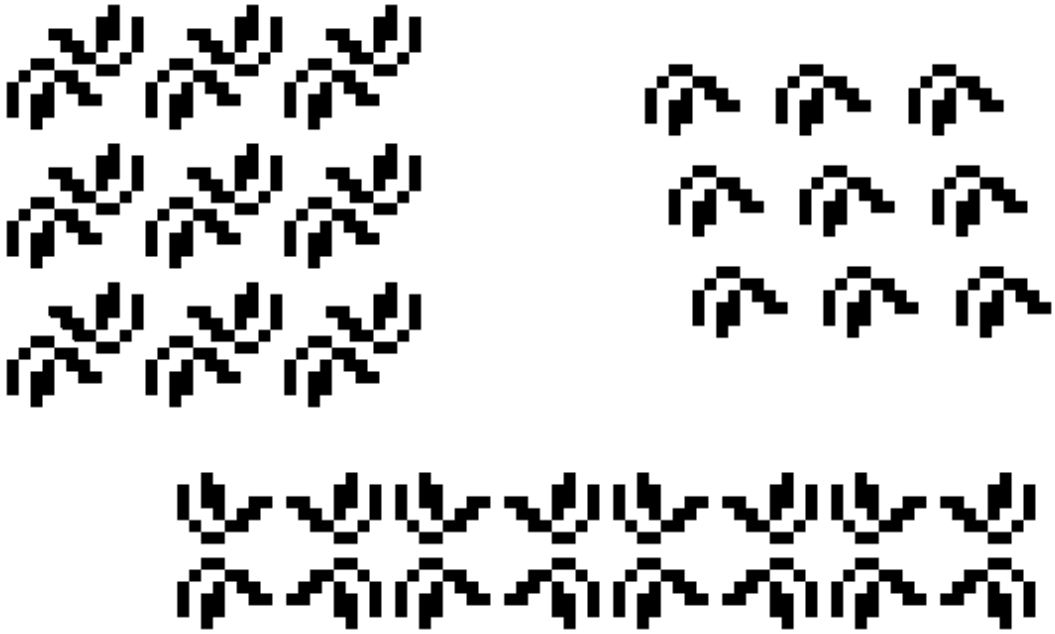


Рис. 2.11. Приклади вишивок груп: p2 - для площини (а); p1 - для площини (б); pmm - для полоси (в).

2.2. Мова опису мінімального рисунку

Показавши, що зображення-орнаменти (вишивка) наділені структурою (підрозділ 1.2) та довівши, що перетворення над зображеннями утворюють групи перетворень (підрозділ 2.1), формалізуємо в даному підрозділі складову зображення-орнаменту - мінімальний рисунок.

Як було показано в підрозділі 1.3, для формалізації мінімального рисунку використаємо структурний підхід. Для граматики (означення граматики приведено в пункті 1.3.2), що застосовується для опису мінімального рисунку вишивки, введемо наступне поняття [10, 24]: виводимість. Якщо є послідовність ланцюжків x_0, x_1,…,x_n, в якій кожен наступний ланцюжок безпосередньо виводиться з попереднього, то ланцюжок x_n виводиться з ланцюжка x_0.

Сукупність всіх ланцюжків, що виводяться з початкового символу в граматиці G, називається мовою, породженою граматикою G і позначається символом L(G) [78, 79].

Мова PDL [79] використана для опису зображень, які будуються за допомогою ліній. Оскільки зображення-орнаменти (вишивки) утворюються в площині, розділеній на квадрати, тому для опису мінімального рисунку запропонована наступна мова опису зображень (для одиниці зображення відведено клітинку (піксел)) [28, 80, 81].

1. Кожен непохідний елемент помітимо в двох різних точках - головній ***г*** і хвостовій ***х*** (рис. 2.12.а). Причому непохідні елементи дотикаються і накладаються тільки в головних чи хвостових точках.

2. Введемо бінарні операції конкатенації (з'єднання):

- операція a + b (рис.2.12.б - головна точка "a" дотикається до хвостової точки "b");

- операція a ⊕ b (рис.2.12.в - головна точка "a" співпадає з хвостовою точкою "b");

- операція a × b (рис.2.12.г - хвостова точка "a" дотикається до хвостової точки "b");

- операція a ⊗ b (рис.2.12.д - хвостова точка "a" співпадає з хвостовою точкою "b");

- операція a - b (рис.2.12.е - головна точка "a" дотикається до головної точки "b");

- операція a Θ b (рис.2.12.є - головна точка "a" співпадає з головною точкою "b");

- операція a ∗ b (рис.2.12.ж - головна точка "a" дотикається до головної точки "b" і хвостова точка "a" дотикається до хвостової точки "b".

3. Введемо наступну унарну операцію:

~ - визначається як переміна головної і хвостової точок (рис. 2.13).

4.Введемо непохідні елементи:

а) a , b , c , d (рис. 2.14.а). Надалі будемо вважати, що $|a|=|b|=|c|=|d|=1$ - ланцюжки довжиною = 1. Якщо a - ланцюжок символів, то a^n - ланцюжок, що складається з n раз побудованого ланцюжка a.

б) "порожні" непохідні елементи:

pa, pb, pc, pd (рис.2.14.б);

в) "несуттєві" непохідні елементи:

na, nb, nc, nd (рис.2.14.в).

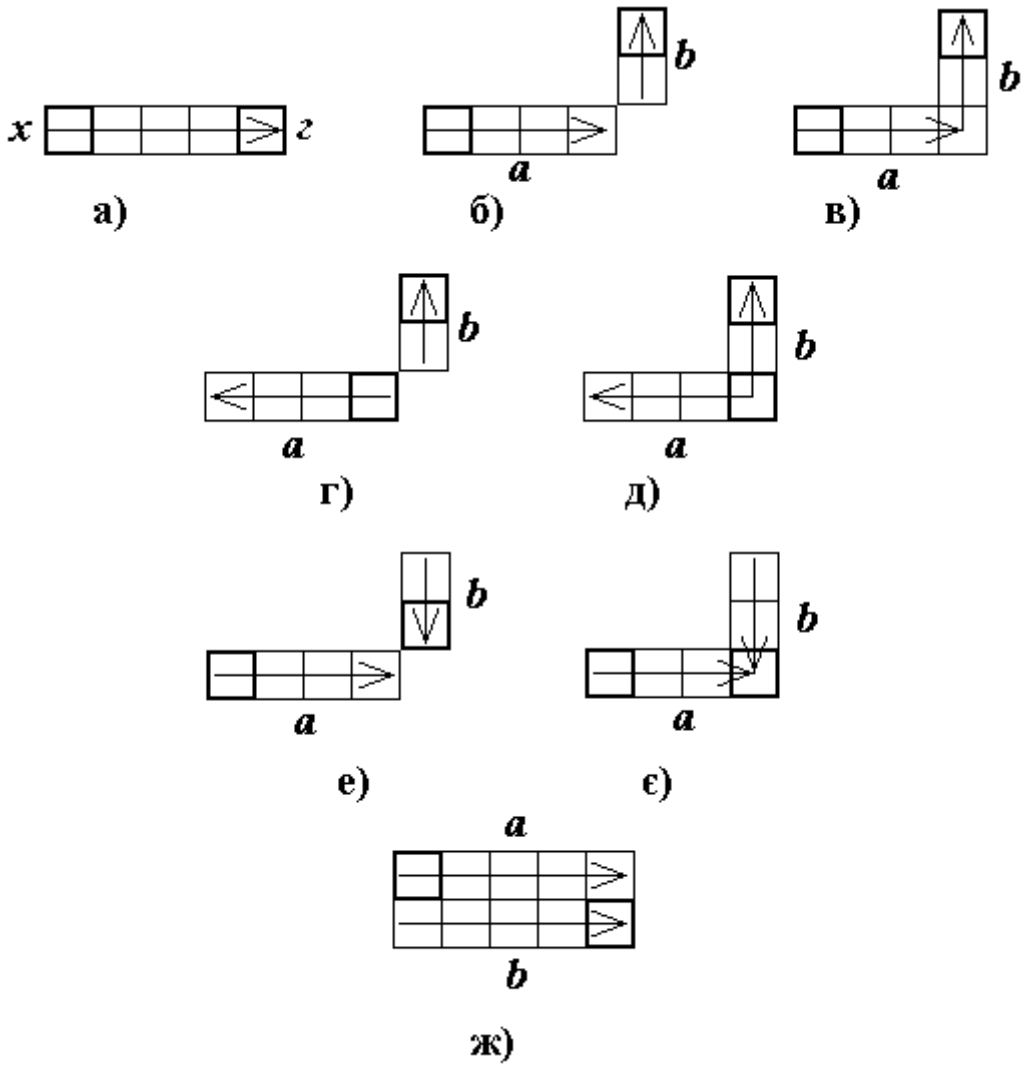


Рис. 2.12. Бінарні операції конкатенації.

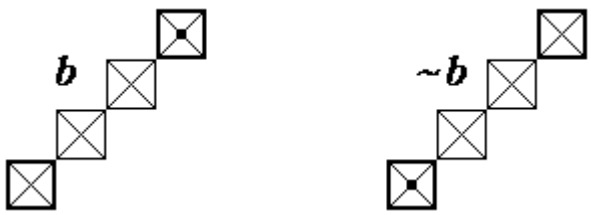


Рис. 2.13. Унарна операція.

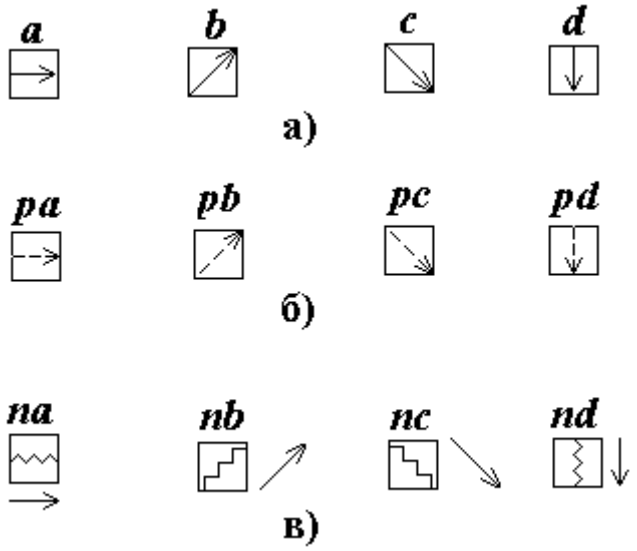


Рис. 2.14. Непохідні елементи.

Непохідні елементи "порожній" і "несуттєвий" з'єднують непересікаючі непохідні елементи і досить корисні для опису простих геометричних зв'язків. "Порожній" використовується при обривах і виступає з'єднуючим "простором" між образами. Коли необхідно описати зв'язок між непохідними елементами, що не перетинаються, а вони є розділені іншими об'єктами, то ці останні об'єкти визначають як несуттєві елементи.

Згідно теореми Гінзбурга і Грейбаха [78] - якщо всі правила заданої граматики G мають вигляд $\varphi \to \phi$, де φ непорожній ланцюжок, а ϕ містить хоча б один термінальний символ, то мова L(G) є безконтекстною. З цього випливає, що запропонована мова опису є безконтекстною.

Граматика, що породжує речення в цій мові є також безконтекстною граматикою.

$V_N = \{S, SF\}, V_T = \{a\} \cup \{+, \oplus, -, \Theta, *, \times, \otimes, (,)\}$,

a - це будь-який непохідний елемент з вище названих,

P : $S \to$ a, $S \to (S \,\&\, S)$, $S \to (\sim S)$, $S \to SF$, $SF \to (SF \,\&\, SF)$, $SF \to (\sim SF)$, $\& \to +$, $\& \to \oplus$, $\& \to -$, $\& \to \Theta$, $\& \to *$, $\& \to \times$, $\& \to \otimes$ [28].

Як семантичну інформацію про непохідні елементи можна розглядати множину ознак, яка потрібна для опису виділеного непохідного елементу. Непохідні елементи з різними властивостями можуть бути описані величинами ознак. Вони повинні містити інформацію, яка є важливою для предметної області. В нашому

випадку важливими є розмір, розміщення, а, отже, в непохідних елементах повинна бути інформація про ці характеристики, щоб класи об'єктів можна було розрізняти при аналізі описів. Притримаємося [24] формальної схеми початкового опису $T(x)$ образу x

$$T(x)=(T_s(x),T_v(x)),$$

де $T_s(x)$ - задання непохідних елементів образу x і їх взаємовідношень, а $T_v(x)$ - значення ознак для кожного елементу. Структурна інформація про образ x задається синтаксичним описом

$$H(x)=(H_s(x),H_v(x)),$$

де $H_s(x)$ - синтаксична компонента (правила підстановки необхідні для породження $T_s(x)$, а $H_v(x)$ - виконання інтерпретаційного правила, що відповідає кожному правилу підстановки граматики G, яке використовується при граматичному розборі виразу $T_s(a)$. Тоді повний опис образу має вигляд:

$$(T(x), H(x))=((T_s(x),T_v(x)), (H_s(x),H_v(x)).$$

Семантична інформація про підобраз (допоміжний символ граматик) оцінюється по семантичній інформації непохідних елементів, що його складають.

Представимо семантичними функціями бінарні операції конкатенації (табл. 2.5). Вираз ДОТ означає "дотикатися до", НАКЛ - "накладатися на".

Для зображення А, показаного на рис. 2.16: $T_s(A) = (\sim(a^3 * a^3)) \Theta (\sim в^2)$;

$H_s(A)$ представлено деревом граматичного розбору (рис. 2.15).

Використаємо формальну семантику Кнута [24] до опису зображень. "Зміст" ланцюжка визначимо через ознаки допоміжного символу, що виникає при його граматичному розборі в відповідності з граматичними правилами. Всі ознаки поділимо на "успадковані" і "синтезовані". Ті ознаки, що визначаються контекстом фрази, назвемо успадкованими, а ті, що зв'язані з самою фразою - синтезовані. Обидва види ознак перебувають в тісному зв'язку. Зміст ланцюжка визначається локальними правилами, які співвідносять ознаки кожного правила підстановки при граматичному розборі ланцюжків. В безконтекстній граматиці кожному правилу підстановки відповідає "семантичне правило", що визначає синтезовані ознаки для символів в лівій частині правила і успадковані для символів в правій частині.

Очевидно, що початковий допоміжний символ не має синтезованих ознак.

Таблиця 2.5

Представлення бінарних операцій семантичними функціями

T_s (B)	Правило підстановки	Семантична функція
$D_1 + D_2$	$B \to D_1 + D_2$	$x(B)=x(D_1)$ $z(B)= z(D_2)$ $z(D_1)$ ДОТ $x(D_2)$
$D_1 \oplus D_2$	$B \to D_1 \oplus D_2$	$x(B)=x(D_1)$ $z(B)= z(D_2)$ $z(D_1)$ НАКЛ $x(D_2)$
$D_1 \times D_2$	$B \to D_1 \times D_2$	$x(B)=x(D_1)$ $z(B)= z(D_2)$ $x(D_1)$ ДОТ $x(D_2)$
$D_1 \otimes D_2$	$B \to D_1 \otimes D_2$	$x(B)=x(D_1)$ $z(B)= z(D_2)$ $x(D_1)$ НАКЛ $x(D_2)$
$D_1 - D_2$	$B \to D_1 - D_2$	$x(B)=x(D_1)$ $z(B)= z(D_2)$ $z(D_1)$ ДОТ $z(D_2)$
$D_1 \Theta D_2$	$B \to D_1 \Theta D_2$	$x(B)=x(D_1)$ $z(B)= z(D_2)$ $z(D_1)$ НАКЛ $z(D_2)$
$D_1 * D_2$	$B \to D_1 * D_2$	$x(B)=x(D_1)$ $z(B)= z(D_2)$ $z(D_1)$ ДОТ $z(D_2)$ $x(D_1)$ ДОТ $x(D_2)$.

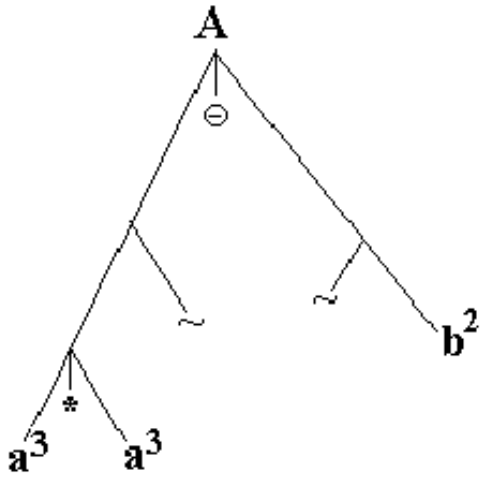


Рис. 2.15. Дерево граматичного розбору

Додамо в безконтекстну граматику семантичні правила наступним чином. Відомо що, скінченна множина ознак відповідає кожному символу $Y \in (V_N \cup V_T)$. Позначимо цю скінченну множину символом B(Y). Згідно вище сказаного, вона поділяється на дві множини $B_0(Y)$ - синтезованих ознак і $B_1(Y)$ - успадкованих. Будемо вимагати, щоб $B_1(S)=\varnothing$, $B_0(Y)=\varnothing$ при $Y \in V_T$. Позначимо через D_z - множину можливих значень ознаки $z \in B(Y)$. З цієї множини при появі символу Y в дереві граматичного розбору може бути вибране одне значення. Припустимо що, множина P складається з m правил підстановки, і нехай *i*-те правило має вигляд

$$Y_{i0} \rightarrow Y_{i1}, Y_{i2}, \ldots, Y_{ini},$$

де $Y_{i0} \in V_N$, $n_i >= 0$ і $Y_{ij} \in (V_N \cup V_T)$, $1 <= j <= n_i$. Семантичні правила - це функції f_{ijz}, що визначені для всіх $1 <= i <= m$, $1 <= j <= n_i$, причому $z \in B_0(Y_{ij})$ при j=0 і $z \in B_1(Y_{ij})$ при j>0. Функції f_{ijz} це відображення прямого добутку $D_{z1} \times \ldots \times D_{zt}$ в множину D_z для деякого $t=t(i,j,z) >= 0$, де кожен $z_p = z_p(i,j,z)$, $1 <= p <= t$, - це ознака деякого символу Y_{ilp} для $0 <= l_p <= n_i$. Отже, кожне семантичне правило переводить певні ознаки символів Y_{i0}, $Y_{i1}, Y_{i2}, \ldots, Y_{ini}$ в величину деякої ознаки символу Y_{ij}.

Визначимо "зміст" ланцюжка безконтекстної мови при допомозі семантичних правил наступним чином. Будується дерево виводу основного ланцюжка *x* з початкового символу S. Для такого дерева S є коренем, а вершини позначаються символами основними або допоміжними. Нехай Y - символ вершини дерева, а $z \in B(Y)$ - ознака цього символу. Коли $z \in B_0(Y)$, то $Y = Y_{i0}$ для деякого *i*, у випадку $z \in B_1(Y)$ - $Y = Y_{ij}$ для деяких *j* і *i*, $1 <= j <= n_i$. Згідно означення, ознака z має величину s для даної вершини, якщо в відповідному семантичному правилі

$$f_{ijz}: D_{z1} \times \ldots \times D_{zt} \rightarrow D_z$$

всі ознаки мають величини $v_1 ,...,v_t$ для відповідних вершин Y_{il1}, , . . . , Y_{ilt}, і v= $=f_{ijz}$ $(v_1 ,...,v_t)$. Оцінювання ознак здійснюється до тих пір, поки не закінчиться дерево. А одержані оцінки ознак для кореня дерева і будуть "змістом", що відповідає даному виводу.

Покажемо застосування формальної семантики до опису зображення А, показаного на рис. 2.16, граматичний розбір приведено в табл. 2.6. В даному випадку образ - це А, (*C(X)* - наявність зв'язності)

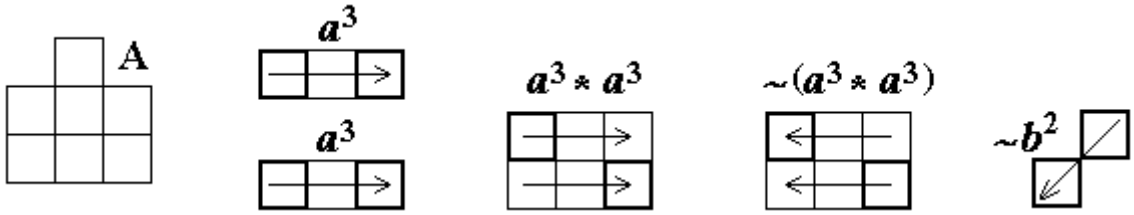


Рис. 2.16. Розклад зображення на непохідні елементи і операції між ними.

Таблиця 2.6

Граматичний розбір

Правило підстановки	Семантичне правило
A1→ $a^3 * a^3$	x(A1)= $x(a^3)_{верхнє}$ $г$(A1) = $г(a^3)_{нижнє}$ C(A1)= $x(a^3)$ДОТ $x(a^3) \wedge г(a^3)$ ДОТ $г(a^3)$
A2 → ~ A1	x(A2)= $г$(A1) $г$(A2)= x(A1) C(A2)= C(A1)
A3 → ~ $в^2$	x(A3) = $г(в^2)$ $г$ (A3) = $x(в^2)$
A → A2 Θ A3	x(A) = x(A2) $г$(A) = $г$(A2)= $г$ (A3) C(A)= $г$(A2) НАКЛ $г$ (A3)∧ C(A2)

Дерево виводу має вигляд рис. 2.17.

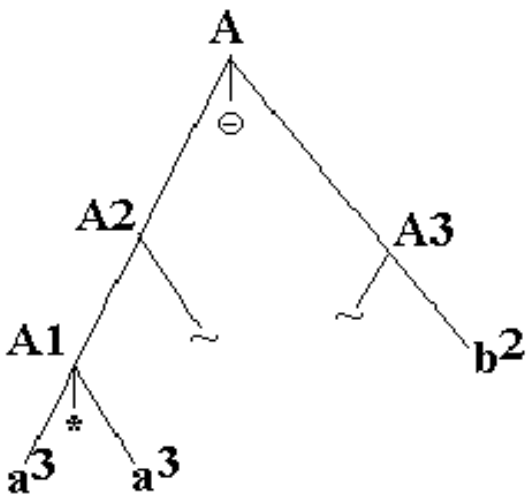


Рис. 2.17. Дерево виводу.

Семантична інформація наступна:

$x(\text{A}) = г(a^3)_{нижнє}$

$г(\text{A}) = x(a^3)_{верхнє} = x(в^2)$

$C(\text{A})=(\, x(a^3)$ ДОТ $x(a^3) \wedge (\, г(a^3)$ ДОТ $г(a^3) \wedge$

$\wedge\, x(a^3)_{верхнє}$ НАКЛ $x(в^2)$

$x(\text{A3}) = г(в^2), \quad г(\text{A3}) = x(в^2)$

$x(\text{A2}) = г(a^3)_{нижнє}$, $г(\text{A2}) = x(a^3)_{верхнє}$

$C(\text{A2}) = C(\text{A1})$

$x(\text{A1}) = x(a^3)_{верхнє}$, $г(\text{A1}) = г(a^3)_{нижнє}$

$C(\text{A1}) = x(a^3)$ДОТ $x(a^3) \wedge г(a^3)$ ДОТ $г(a^3)$.

Визначено алгебру, яка буде складатися з деякої непустої множини непохідних елементів - множина Е і послідовності визначених на Е операцій F={~, +, ⊕ , -, Θ , ∗ , × , ⊗}.

Для даної алгебри виконуються наступні закони:

1. Комутативності - справджується для наступних операцій (але тільки для побудови, початкова і кінцева точки не співпадають).

а) ∗

$(U_1 * U_2)=(U_2 * U_1)$;

б) ×, ⊗

$(U_1 \times U_2)=(U_2 \times U_1)$;

$(U_1 \otimes U_2)=(U_2 \otimes U_1)$;

в) -, Θ

$(U_1 - U_2) =(U_2 - U_1)$;

$(U_1 \Theta U_2)=(U_2 \Theta U_1)$.

2. Асоціативності

а) $((U_1 + U_2)) + U_3) = (U_1 + (U_2 + U_3))$;

б) $((U_1 \oplus U_2)) \oplus U_3) = (U_1 \oplus (U_2 \oplus U_3))$.

3. Інверсії

$(\sim (\sim U)) = U$.

4. Для операції ~ справджуються наступні рівності:

$(\sim(U_1 + U_2)) = ((\sim U_2) + (\sim U_1))$;

$(\sim(U_1 \oplus U_2)) = ((\sim U_2) \oplus (\sim U_1))$;

$(\sim(U_1 * U_2))_! = ((\sim U_2) * (\sim U_1))$;

$(\sim(U_1 \times U_2)) = ((\sim U_2) - (\sim U_1))$;

$(\sim(U_1 \otimes U_2)) = ((\sim U_2)\ \Theta\ (\sim U_1))$; (2.1)

$(\sim(U_1 - U_2)) = ((\sim U_2) \times (\sim U_1))$;

$(\sim(U_1\ \Theta U_2)) = ((\sim U_2) \otimes (\sim U_1))$. (2.2)

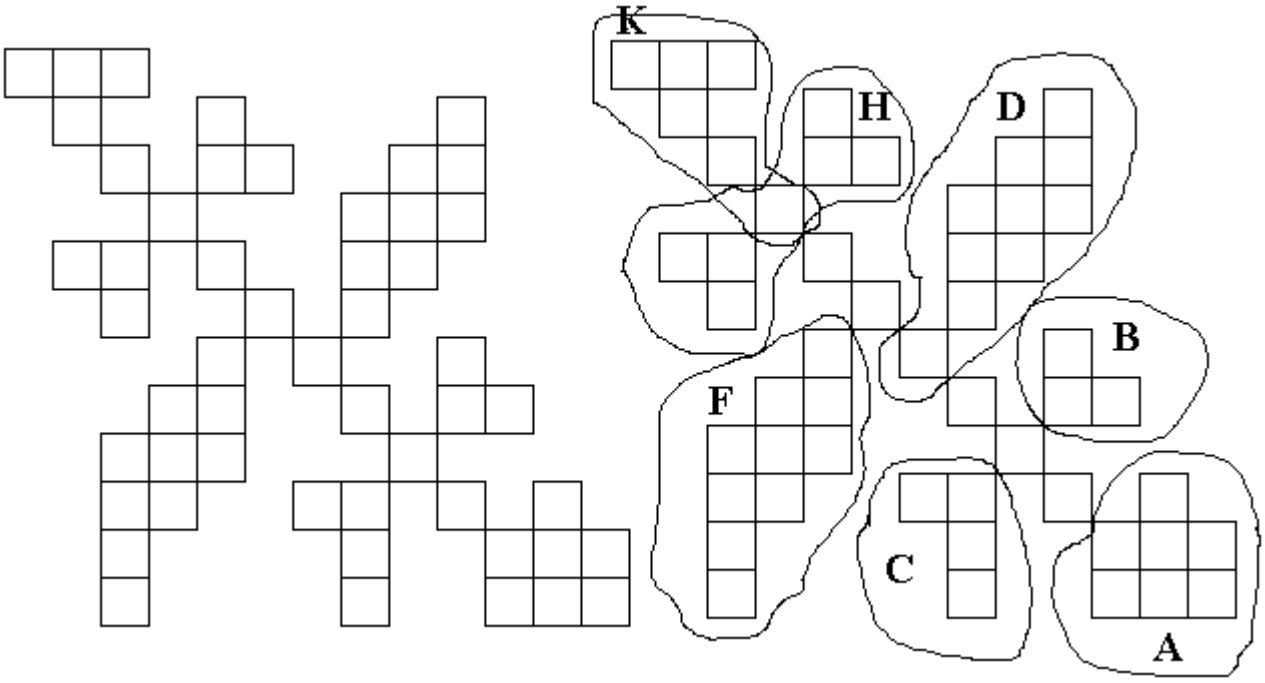
Справедлива наступна теорема:

Теорема 1. Будь-який ланцюжок з'єднань описує одне єдине зображення.

Доведення.

Доведення маємо з того, що одне і тільки одне з'єднання непохідних елементів представляється шляхом застосування мови опису зображень.

Розглянемо застосування даної граматики до зображення *S* на рис.2.18. Вихідні елементи *A, B, C, D, F, K, H* показані на рис.2.19.



а) б)

Рис. 2.18. Зображення S: вигляд зображення (а); розбиття на складові (б).

$S \rightarrow ((c^2+(c^2-A \times B \times C)) \otimes D \otimes(\sim c^2) + F \otimes(\sim c^4) \otimes H \otimes K$.

$A \to (\sim(a^3 * a^3)) \Theta (\sim b^2)$;

$B \to a^2 \otimes (\sim d^2)$;

$C \to (\sim a^2) \otimes d^3$;

$D \to b^2 \oplus ((\sim d^3) \oplus b^2 \oplus d^3 \oplus b^2 \oplus (\sim d^3)$;

$F \to d^3 \oplus (\sim b^2) \oplus (\sim d^3) \oplus (\sim b^2) \oplus d^4$;

$K \to (\sim c^4) \oplus a^3$;

$H \to (\sim b^2) \oplus (d^2 \otimes (\sim a^2)) \times B$.

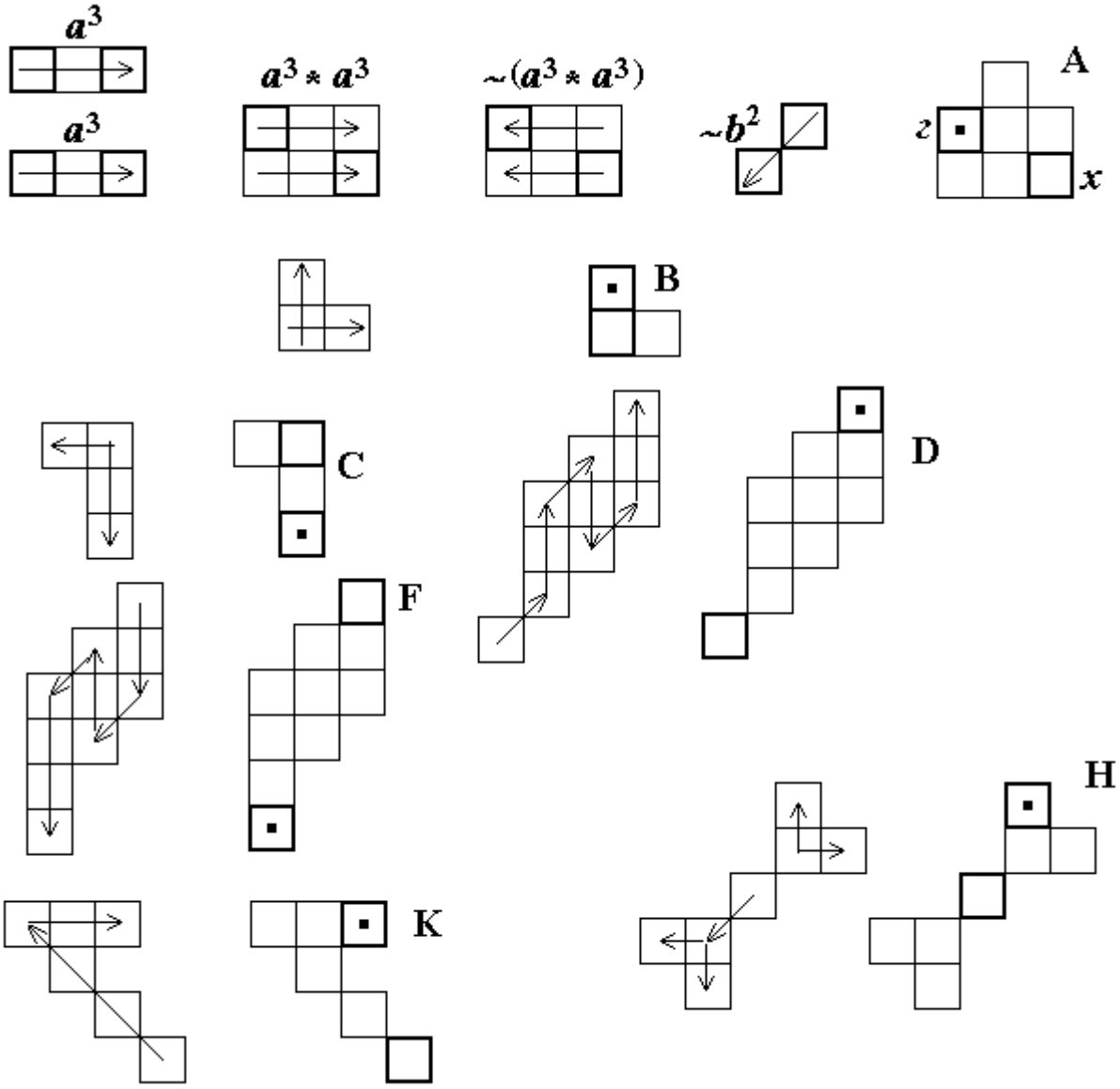


Рис.2.19. Складові зображення S.

Таким чином, в даному підрозділі запропоновано мову опису зображень мінімального рисунку, досліджено її властивості, що дало можливість синтезувати

мінімальні рисунки і на цій основі здійснити моделювання складних зображень-орнаментів.

2.3. Формалізація рапортів груп перетворень

Результати, отримані в підрозділах 2.1, 2.2 дозволяють описувати групи перетворень зображень та їх мінімальні рисунки. Третьою складовою розробки методу опису та моделювання є формалізація рапортів (компонентів груп перетворень). Для опису рапортів і підорнаментів використаємо рекурсивний підхід, обгрунтування якого приведено в пункті 1.3.2.

В математиці широко використовується рекурсивна схема так звана примітивна рекурсія [14, 64]:

$$\left.\begin{aligned} f(0) &= b \\ f(n) &= h(n, f(n-1)) \end{aligned}\right\}, \qquad (2.3)$$

де b - натуральне число; f, h - функції, значення яких теж є натуральними числами. Наприклад:

$b \times 1 = b,$ $\qquad$ $0! = 1,$

$b \times n = b \times (n - 1) + b,$ $\qquad$ $n! = n \times (n - 1)!.$

Побудуємо схему (2.3) не тільки для числових функцій, а й для об'єктів іншого виду (природи) [14]. Нехай B - довільний об'єкт класу Ψ, $B \in \Psi$, F - функція чи оператор, аргументом якої є натуральне число, а результатом - об'єкт з Ψ; H - функція 2-х аргументів натурального числа і об'єкту з Ψ, а результатом - об'єкт з Ψ, тоді рекурсивним визначенням F на класі Ψ буде наступне

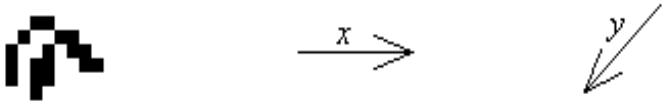
$$\left.\begin{aligned} F(0) &= B \\ F(n) &= H(n, F(n-1)) \end{aligned}\right\}. \qquad (2.4)$$

Заміна числа b на об'єкт B значно розширяє область застосування рекурсивної схеми. При схемі (2.3) породжується тільки лінійка чисел, а при схемі (2.4) складні об'єкти, що мають ієрархічну структуру. Рекурсивне визначення дає можливість в компактному виді дати опис для деякого об'єкта чи групи об'єктів. Застосуємо його до груп перетворень над складними зображеннями.

Дослідження велися для 12-ти федорівських груп перетворень на площині Z = {*p1, p2, pm, pg, cm, pmm, pmg, pgg, cmm, p4, p4m, p4g*} і семи на полосі J = {*p1, p2, pm, pg, cm, pmm, pmg, pm, pmm*} [31, 81] .

Розглянемо групу перетворень *p1*, що складається лише з паралельних переносів, дещо з іншої точки зору, ніж в підрозділі 2.1. Для побудови її достатньо задати елементарну частину (мінімальний рисунок) і переноси (правило перетворення елементарної частини), які потім розповсюджуються «в ширину» - на всі елементи і «в глибину» - на інші рівні.

Нехай мінімальний рисунок має вигляд рис. 2.20, а переноси здійснюються в напрямку X та Y. Тоді такт рекурсії має вигляд рис. 2.21.



а) б)

Рис. 2.20. Мінімальний рисунок (а); напрямки переносів (б).

Рис. 2.21. Один такт рекурсії.

Повторяючи процес 4 такти одержимо рис. 2.22.

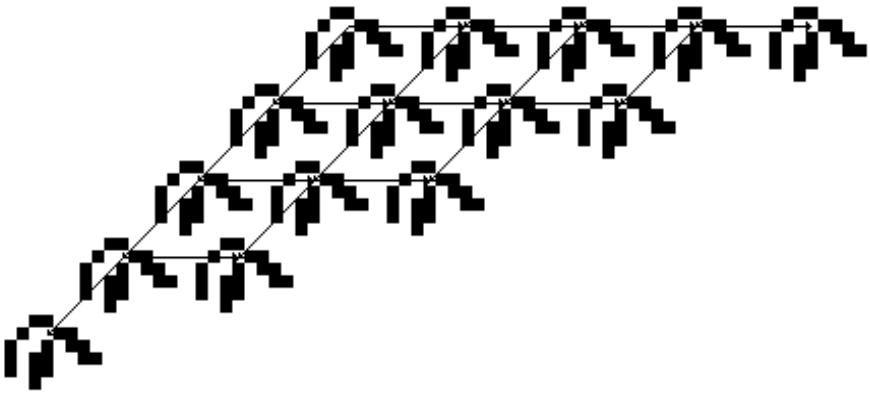


Рис. 2.22. 4 такти рекурсії.

По схемі (2.4) визначимо групу перетворень *p1* наступним чином [31, 81]:

$$\left.\begin{array}{l} \textit{група } p1(0) = D\min \\ \textit{група } p1(n) = \textit{група } p1(n-1) \textit{ над} \\ \textit{якою зроблені переноси в напрямку} \\ \qquad X \textit{ та } Y \end{array}\right\} \qquad (2.5)$$

Оператор H (переноси в напрямку *X* та *Y*) дає можливість щоб за один такт рекурсії на *n* - му кроці проходило паралельно декілька елементарних змін в об'єкті F *(n -1)*.

Приведемо кожну з розглянених вище 11 груп до схеми (2.5), представивши геометричні викладки (тобто покажемо, що кожна з них є підгрупою *p1*-групи переносів), домовившись, що знак «+» означає об'єднання зображень, а *f*(W)-перетворене зображення (об'єкт) над зображенням W, де *f* є {*R, R_1, R_2, P, Q, X, Y, T, S*}, *R, R_1, R_2* - відображення, *P, Q* - ковзні відображення, *X* , *Y* - переноси, *T*, *S* - повороти на 180° та 90°.

Зображення мінімального рисунка повертаємо на 180° відносно О. Об'єднання мінімального і перетвореного дасть елементарний рисунок, який при перенесенні в напрямках *X* та *Y* побудує групу *p2* (рис.2.23). На мові мистецтвознавців цей елементарний рисунок називається рапортом [37]. Отже, для *p2* буде дійсною схема

$$\left.\begin{array}{l} \textit{група } p2(0) = D\min + T(D\min) \\ \textit{група } p2(n) = \textit{група } p2(n-1) \textit{ над} \\ \textit{якою зроблені переноси в напрямку} \\ \qquad X \textit{ та } Y \end{array}\right\} \qquad (2.6)$$

Примітка: надалі буде приводитися лише перша тотожність рекурсивних схем, друга буде для всіх груп аналогічною, лише буде змінюватися позначення групи.

Мінімальний рисунок відображаємо відносно R_1. Об'єднання мінімального рисунку і утвореного (рис. 2.24) дадуть рапорт для групи *pm*. Тоді

$$\textit{група } pm(0) = D\min + R_1(D\min).$$

Рис. 2.23. Рапорт групи *p2***.**

Рис. 2.24. Рапорт групи *pm***.**

Над мінімальним рисунком здійснюємо ковзне відображення Q (рис. 2.25), а потім над об'єднанням цих двох рисунків проводимо ще одне ковзне відображення P. Якщо все утворене зображення будемо переносити в напрямку X та Y то побудуємо групу *pg*. Отже,

$$\textit{група } pg(0) = D\min + Q(D\min) + P(D\min + Q(D\min)).$$

Мінімальний рисунок відображаємо відносно R. Об'єднання мінімального рисунку і утвореного (рис. 2.26) дадуть рапорт для групи *cm*. Тоді

$$\textit{група } cm(0) = D\min + R\ (D\min).$$

Якщо мінімальний рисунок відобразити відносно осі R_1 (рис. 2.27), об'єднати ці два рисунки, потім це об'єднання відобразити відносно R_2 (R_1 і R_2 є перпедикулярними між собою), то одержимо перше рекурсивне співвідношення для групи *pmm*:

$$\textit{група } pmm(0) = D\min + R_1(D\min) + R_2(D\min + R_1(D\min))$$

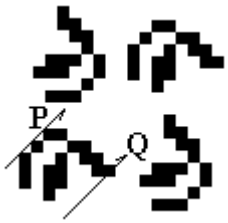


Рис. 2.25. Рапорт групи *pg*.

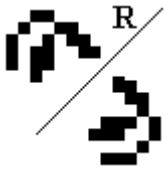


Рис. 2.26. Рапорт групи *cm*.

Над мінімальним рисунком здійснюємо поворот на 90° відносно O (рис. 2.28), а потім над об'єднанням цих двох рисунків проводимо відображення R. Якщо все утворене зображення будемо переносити в напрямку X та Y (X та Y не під прямим кутом) то побудуємо групу *pmg*. Отже,

$$\textit{група } pmg(0) = D\min + T(D\min) + R\ (D\min + T(D\min)).$$

Якщо над мінімальним рисунком будемо здійснювати ковзне відображення Q (рис. 2.29), а потім над об'єднанням цих двох рисунків будемо проводити ще одне ковзне відображення P (обов'язковою умовою є перпендикулярність Q і P) і якщо все утворене зображення будемо переносити в напрямку X та Y то побудуємо групу *pgg*. Отже,

$$\textit{група } pgg(0) = D\min + Q(D\min) + P(D\min + Q(D\min)).$$

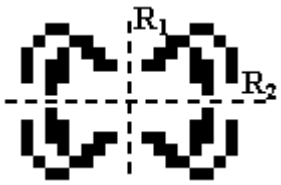


Рис. 2.27. Рапорт групи *pmm***.**

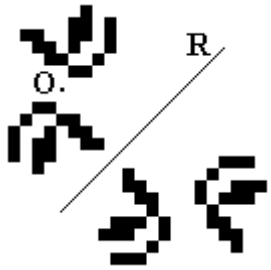


Рис. 2.28. Рапорт групи *pmg***.**

Якщо мінімальний рисунок відобразити відносно осі R_1 (рис. 2.30), об'єднати ці два рисунки, потім це об'єднання відобразити відносно R_2, то одержимо перше рекурсивне співвідношення для групи *cmm***:**

$$\textit{група } cmm(0) = D\min + R_1(D\min) + R_2(D\min + R_1(D\min)).$$

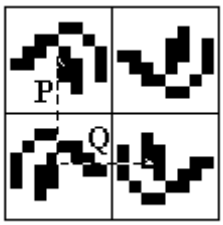


Рис. 2.29. Рапорт групи *pgg***.**

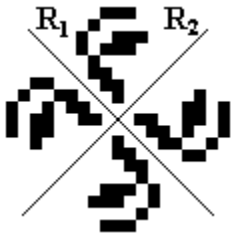


Рис. 2.30. Рапорт групи *cmm***.**

Мінімальний рисунок повернемо на 90° відносно деякої точки O (рис. 2.31), потім утворений рисунок повертаємо на 90° і, нарешті, останній утворений рисунок повертаємо знову на 90°. Об'єднання мінімального і трьох утворених буде становити рапорт для групи *p4*, або іншими словами для цієї групи буде виконуватися рекурсивна схема

$$\textit{група } p4(0) = D\min + S(D\min) + S(S(D\min)) + S(S(S(D\min))).$$

Цю ж групу можна було утворити дещо інакше: мінімальний рисунок повернемо на 90° відносно деякої точки O, потім об'єднання вихідного і утвореного повертаємо на 180°. Рекурсивна схема в цьому випадку буде наступна:

$$\textit{група } p4(0) = D\min + S(D\min) + T(D\min + S(D\min)).$$

Якщо мінімальний рисунок відобразимо відносно R, як показано на рис. 2.32, потім об'єднання вихідного і утвореного відображаємо відносно R_1, нарешті

об’єднання чотирьох рисунків відображаємо відносно R_2. Неважко бачити, що

$$\textit{група } p4m(0) = D\min + R\,(D\min\,) + R_1(D\min + R\,(D\min\,)) + R_2(D\min +$$

$$+ R\,(D\min\,) + R_1(D\min + R\,(D\min\,))$$

Рис. 2.31. Рапорт групи *p4*.

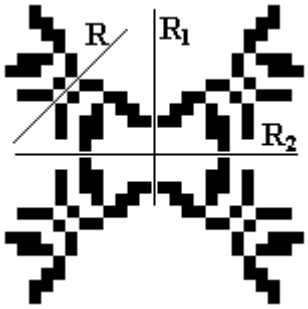


Рис. 2.32. Рапорт групи *p4m*.

Над мінімальним рисунком будемо здійснювати відображення R (рис. 2.33), потім об’єднання вихідного і утвореного рисунків повертаємо на 90° відносно деякої точки О. Далі утворені рисунки повертаємо на 90° і, нарешті, останні утворені рисунки повертаємо знову на 90°. Об’єднання мінімального і всіх утворених буде становити рапорт для групи *p4g*, або іншими словами для цієї групи буде виконуватися рекурсивна схема

$$\textit{група } p4g(0) = D\min + R(D\min\,) + S(D\min + R(D\min)) + S(S(D\min +$$

$$+ R(D\min))) + S(S(S(D\min + R(D\min))))$$

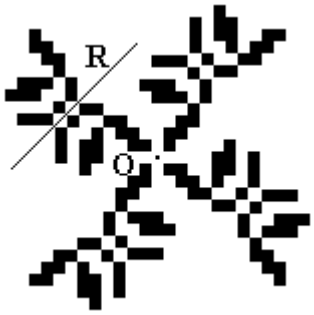


Рис. 2.33. Рапорт групи *p4g*.

Цю ж групу можна утворити дещо інакше: мінімальний рисунок відображаємо відносно R, об’єднання вихідного і утвореного рисунків повертаємо на 90° відносно деякої точки О, потім об’єднання вихідного і трьох утворених рисунків повертаємо на 180°. Рекурсивна схема наступна:

$$\textit{група } p4g(0) = D\min + R(D\min) + S(D\min + R(D\min)) + T(D\min + $$

$$+ R(D\min) + S(D\min + R(D\min)))$$

По аналогії побудуємо рекурсивні схеми для груп на смузі. Для *p1* на смузі рекурсивна схема буде мати вигляд:

$$\left.\begin{array}{l} \textit{група } p1(0) = D\min \\ \textit{група } p1(n) = \textit{група } p1(n-1) \textit{ над} \\ \textit{якою зроблено перенос в напрямку } X \end{array}\right\} \qquad (2.7)$$

Нехай мінімальний рисунок має вигляд рис.2.20.а, а перенос здійснюються в напрямку X (рис. 2.20.б).

За один такт рекурсії одержимо рис. 2.34. За 4-и такти рис. 2.35.

Рис. 2.34. Один такт рекурсії.

Рис. 2.35. Чотири такти рекурсії.

Покажемо, аналогічно що кожна з 7 груп на смузі є підгрупою групи *p1*, отже, вона може зображатися схемою (2.7).

Рапорт групи *pg* утворюється поєднанням мінімального рисунку і утвореного ковзним відображенням P (рис. 2.36). Рекурсивна схема матиме вигляд:

$$\textit{група } pg(0) = D\min + P(D\min)$$

Якшо мінімальний рисунок відобразити відносно осі R_1, як показано на рис. 2.37, то об'єднання цих двох рисунків дасть рапорт групи p1m, а перше співвідношення рекурсивної схеми буде наступним:

$$\textit{група } p1m(0) = D\min + R_1(D\min)$$

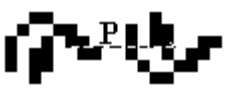


Рис. 2.36. Рапорт групи *pg*.

Рис. 2.37. Рапорт групи *p1m*.

Зображення мінімального рисунка повертаємо на 180° відносно О. Об'єднання мінімального і перетвореного дасть елементарний рисунок, який при перенесенні в

напрямку *X* побудує групу *p2* (рис. 2.38). Отже, для *p2* буде дійсною схема

$$\textit{група } p2(0) = D\min + T(D\min).$$

Якшо мінімальний рисунок відобразити відносно осі R_1, як показано на рис.2.39, потім об'єднання цих двох рисунків повернути на кут 180° відносно деякої точки О, то одержимо рапорт групи *pmg*, а перше співвідношення рекурсивної схеми буде наступним:

$$\textit{група } pmg(0) = D\min + R_1(D\min) + T(D\min + R_1(D\min)).$$

Рис. 2.38. Рапорт групи *p2***.**

Рис. 2.39. Рапорт групи *pmg***.**

Якщо мінімальний рисунок відобразити відносно осі R (рис. 2.40), об'єднати ці два рисунки, то одержимо перше рекурсивне співвідношення для групи *pm:*

$$\textit{група } pm(0) = D\min + R(D\min).$$

Якщо рапорт групи *pm* відобразити відносно осі R, як показано на рис 2.41, то одержимо рапорт групи *pmm*. Звідси справедливою буде така схема:

$$\textit{група } pmm(0) = D\min + R(D\min) + R_1(D\min + R(D\min)).$$

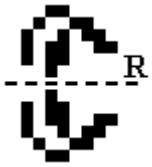


Рис. 2.40. Рапорт групи *pm***.**

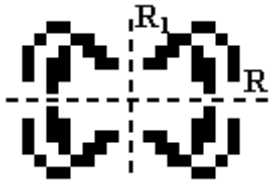


Рис. 2.41. Рапорт групи *pmm.*

Рекурсивні схеми (2.5), (2.7) дають можливість, якщо відомий мінімальний рисунок, одержати будь-яку групу зображень. Вони описують тільки структуру окремого об'єкта, яку можна зображати в вигляді графа $\Omega(\Gamma, \Psi)$ [82], де Γ - множина вершин, Ψ - множина дуг. Дуга графа відповідає застосуванню оператора Н, вершина - один із множини можливих варіантів підстановок конкретних об'єктів. Конкретніше, дуга графа - це переноси в напрямку X та в напрямку Y для схеми (2.5), в напрямку X для схеми (2.7). Вершина - рапорт (РП) в позначеннях на

графі (рис. 2.42).

Таким чином, запропонований у підрозділі 2.3 рекурсивний підхід до опису груп перетворень дозволив формалізувати рапорти, що дало можливість розробити алгоритми синтезу підорнаментів.

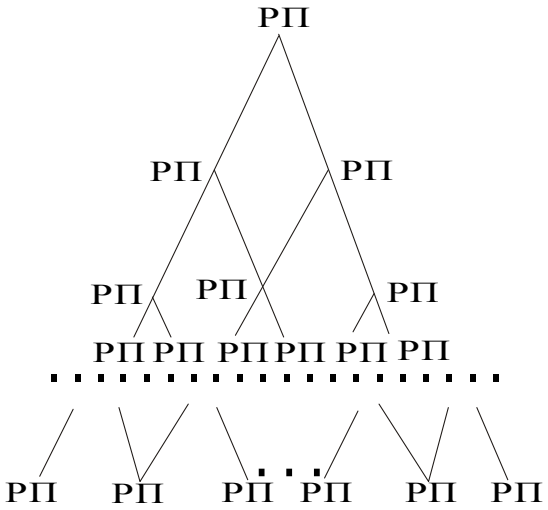


Рис.2.42. Структура груп площини.

2.4. Основні результати, отримані у розділі 2

1. Показано, що перетворення над зображеннями-орнаментами утворюють 12 груп перетворень на площині та 7 груп перетворень на смузі, які входять до відповідних федорівських груп, що дало можливість їх формалізувати.
2. Запропоновано мову опису мінімального рисунку, досліджено її властивості, що дало можливість синтезувати мінімальні рисунки і провести їх моделювання.
3. Запропонований рекурсивний підхід до опису груп перетворень дозволив формалізувати рапорти, та розробити алгоритми синтезу підорнаментів.
4. На основі запропонованої мови опису мінімального рисунку і груп перетворень здійснено моделювання складних зображень-орнаментів

РОЗДІЛ 3
СИНТЕЗ АЛГОРИТМІВ ПОБУДОВИ СКЛАДНИХ ЗОБРАЖЕНЬ-ОРНАМЕНТІВ ТА ЇХ МОДЕЛЮВАННЯ

3.1. Алгоритми побудови груп перетворень на базі породжуючих перетворень

Розробивши в другому розділі метод опису та моделювання складних зображень-орнаментів синтезуємо в даному розділі алгоритми побудови груп перетворень, алгоритми побудови мінімального рисунку та узагальнений алгоритм синтезу зображень- орнаментів.

Розглянемо складні зображення-орнаменти на площині і на смузі. Синтез і моделювання всіх зображень-орнаментів здійснимо на базі мінімальних рисунків (Dmin) і породжувальних перетворень над ними. До останніх належать: відображення, поворот, паралельний перенос, ковзне відображення.

Нижче запропоновано новий підхід до груп перетворень на базі породжувальних перетворень. Для цього розглянемо прямокутно-решітчату сітку, що складається з прямокутників з сторонами x', y'. Алгоритми перетворень приведемо в вигляді матриць перетворень. В загальному випадку матриця перетворень порядку 3×3, матиме вигляд $\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$ [83-87], де a, b, c і d здійснюють зсув, поворот; m і n виконують зміщення, а p і q - одержання проекцій. Елемент s проводить повну зміну масштабу. При $s < 1$ проходить збільшення, а при $s > 1$ зменшення масштабу [85]. З метою одержання загальних результатів будемо використовувати композицію перетворень. Основною перевагою об’єднаних перетворень є те, що ефективніше застосувати одне результуюче перетворення, ніж декілька поетапних.

Побудуємо матриці перетворень для груп множини J [88, 89]. Група *p1* - складається тільки з переносів (рис. 3.1). Всі точки мінімального рисунку піддаються

перетворенню, матрицю якого (в однорідних координатах) можна записати у вигляді:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix}.$$

pg - складається з ковзного відображення відносно осі OX (рис. 3.2). Результуюча матриця наступна:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ x' & 0 & 1 \end{bmatrix}.$$

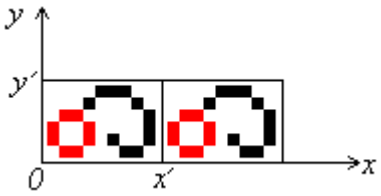


Рис. 3.1. Група *p1*, породжена переносом.

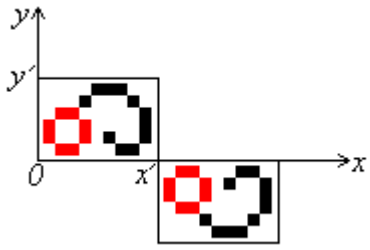


Рис. 3.2. Група *pg*, породжена ковзним відображенням.

У випадку, коли вісь ковзного відображення *l* не співпадає з віссю OX, переносимо вісь *l* на OX, проводимо перетворення, а потім переносимо вісь *l* в вихідне положення. Результуюча матриця матиме вигляд:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -y_1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & y_1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ x' & 2y_1 & 1 \end{bmatrix}.$$

Результати перетворень приведено на рис. 3.3.

p1m - складається з двох відображень: 1-е (R_1) - відносно осі *OY*, 2-е (R_2) - відносно прямої паралельної *OY* і що проходить через точку (x', 0). В результаті отримаємо наступні матриці перетворень:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 2x' & 0 & 1 \end{bmatrix}.$$

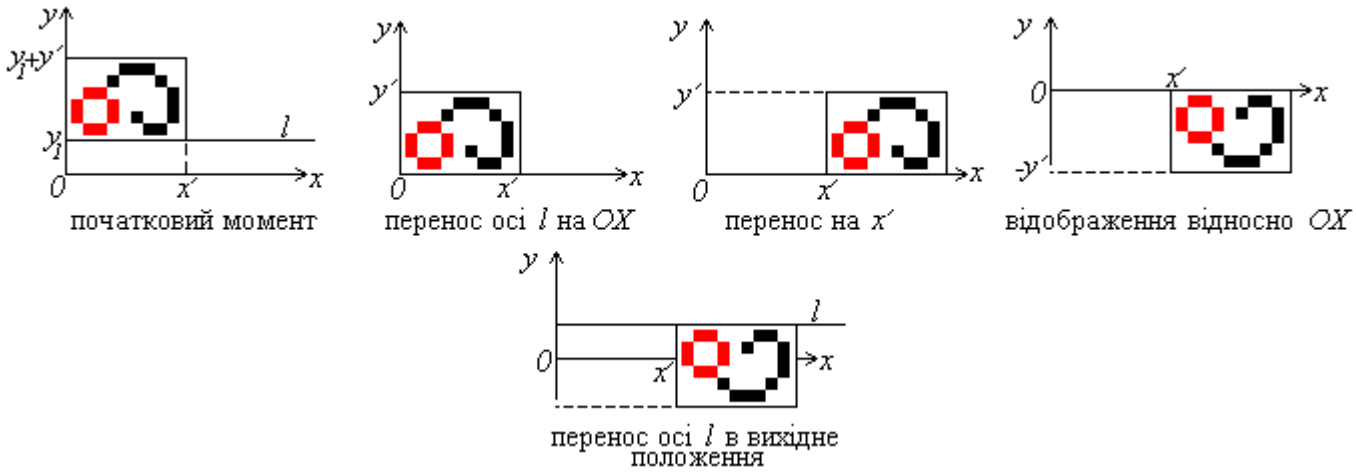


Рис. 3.3. Ковзне відображення відносно осі *l*.

Результати перетворень приведено на рис 3.4.

p2 - складається з 2-х поворотів на 180° відносно (0, 0) і відносно (*x*′, 0). Відзначимо, що матриця повороту на кут θ відносно початку координат має вигляд $\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Поворот відносно довільної точки *M* здійснюється в три етапи:

1) перенос, при якому точка *M* переміщається в початок координат;
2) поворот;
3) перенос, при якому точка *M* з початку координат повертається в початкове положення.

Враховуючи вище сказане, результуючі перетворення будуть наступні:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ x' & 0 & 1 \end{bmatrix}$$

Результат зображений на рис. 3.5.

Зауваження: Центрами поворотів не завжди є точки (0, 0), (x', 0). Ними може бути довільна точка прямих x=0, x=x'.

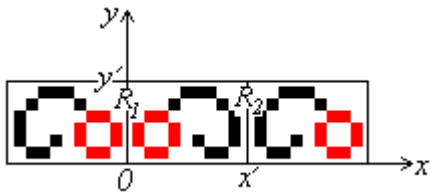


Рис. 3.4. Група *p1m*, породжена двома відображеннями.

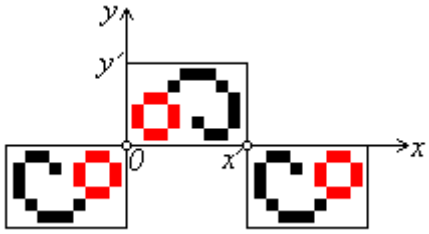


Рис. 3.5. Група *p2*, породжена двома поворотами на 180°.

pmg - складається з відображення відносно *OY* і повороту на 180° відносно (x',0). Дві результуючі матриці перетворень наступні:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ x' & 0 & 1 \end{bmatrix}.$$

Результат перетворень зображений на рис. 3.6.

Зауваження: Центром повороту не завжди є точка (x', 0). Ним може бути довільна точка прямої x=x'.

pm - складається з переносу вздовж осі OX (1-а результуюча) і здійснюється відображення відносно R=OX (2-а результуюча):

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Результат перетворень зображений на рис.3.7.

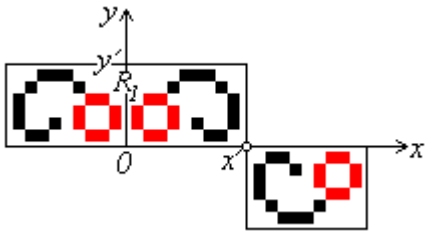


Рис. 3.6. Група *pmg*, породжена відображенням і поворотом на 180°.

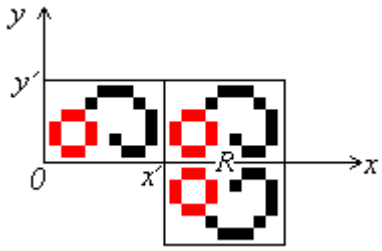


Рис. 3.7. Група *pm*, породжена переносом і відображенням.

Зауваження: вісь R не завжди співпадає з OX. Якщо це так тоді, переносимо вісь R на OX, проводимо перетворення, а потім переносимо вісь R в вихідне положення.

pmm - складається з 3-ох результуючих матриць перетворень. 1-а - відображення відносно $R_1 = OY$, 2-а - відображення відносно R_2=OX, 3-я - відображення відносно $R_3 = x'$:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 2x' & 0 & 1 \end{bmatrix}.$$

Зауваження аналогічне як і для *pm* стосовно R_2. Результат перетворень приведений на рис. 3.8.

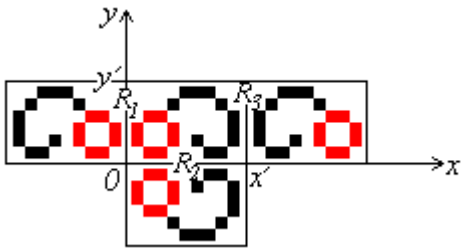


Рис. 3.8. Група *pmm*, породжена трьома відображенями.

Аналогічним чином розглянемо групи перетворень з множини Z (групи перетворень на площині) [88, 89].

p1 - складається тільки з переносів. Всі точки мінімального рисунку піддаються перетворенням, матриці яких можна записати у вигляді: (3.1) - перенос на величину x'; (3.2) - перенос на величину y' (рис. 3.11).

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & y' & 1 \end{bmatrix} \tag{3.2}$$

Зауваження: часто переноси здійснюються не строго в перпендикулярних напрямках, а під будь-яким кутом.

p2 - складається з трьох поворотів на 180° відносно середин сторін трикутника ABO (рис. 3.10). З результуючі матриці перетворень матимуть вигляд (3.3 - 3.5). Результат перетворень приведений на рис. 3.10.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{2}x' & -\frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2}x' & \frac{1}{2}y' & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ x' & y' & 1 \end{bmatrix} \tag{3.3}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2}y' & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & y' & 1 \end{bmatrix} \tag{3.4}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{2}x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2}x' & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ x' & 0 & 1 \end{bmatrix} \tag{3.5}$$

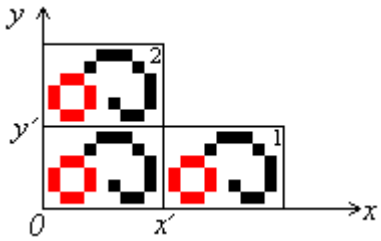


Рис. 3.9. Група *p1*, породжена двома переносами.

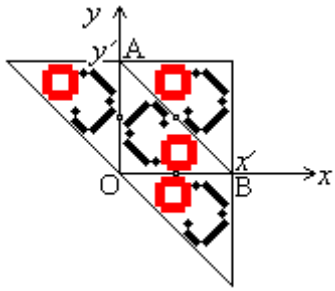


Рис. 3.10. Група *p2*, породжена трьома поворотами.

pm - складається з наступних перетворень. 1-е - перенос елементарного рисунку на величину y'; 2-е - відображення відносно OY; 3-є відображення відносно $x=x'$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & y' & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 2x' & 0 & 1 \end{bmatrix}.$$

Результат перетворень зображений на рис. 3.11.

pg - будується при $y'=x'$ і складається з двох ковзних відображень, осі яких паралельні прямій $y=x$, крок рівний половині діагоналі квадрата. Схема побудови наступна:

1) перенос, при якому вісь ковзного відображення переміщається на пряму $y=x$ (точка $(\frac{1}{2}x';0)$ з початком координат); 2) відображення відносно $y=x$; 3) перенос на величину $(\frac{1}{2}x';\frac{1}{2}y')$; 4) Перенос, при якому вісь ковзного відображення повертається в початкове положення. Матриці перетворень і їх результуюча матимуть вигляд:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{2}x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2}x' & \frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2}x' & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ x' & \frac{1}{2}y' - \frac{1}{2}x' & 1 \end{bmatrix}.$$

Аналогічно і для другого ковзного відображення досить змістити середину сусідньої

сторони $(0;\ \frac{1}{2}y')$ в початок координат:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2}x' & \frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2}y' & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ \frac{1}{2}x' - \frac{1}{2}y' & y' & 1 \end{bmatrix}.$$

Результат перетворень приведений на рис. 3.12.

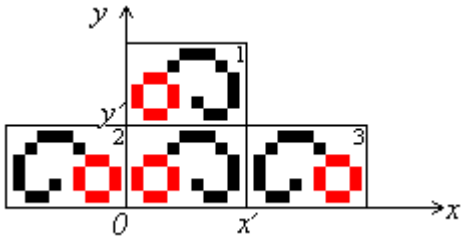


Рис. 3.11. Група *pm*, породжена переносом і двома відображеннями.

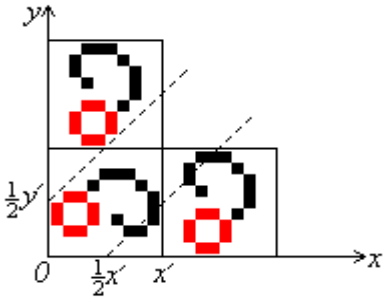


Рис. 3.12. Група *pg*, породжена двома паралельними ковзними відображеннями.

cm - будується при $y'=x'$ і складається з відображення відносно $y=x$ (R) і ковзного відображення. Вісь цього відображення паралельна прямій $y=x$, крок рівний половині діагоналі квадрата. Дві результуючі матриці перетворень мають вигляд:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2}x' & \frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2}y' & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ \frac{1}{2}x' - \frac{1}{2}y' & y' & 1 \end{bmatrix}.$$

Результат перетворень приведений на рис. 3.13.

pmm - складається з 4-ох відображень відносно сторін прямокутника з сторонами, що лежать на OX, OУ, $y = y'$, $x = x'$. Отже, є 4 матриці перетворень: 1-а

- відносно OX: (3.6); 2-а - відносно OУ: (3.7); 3-я відносно сторони $y = y'$: (3.8); 4-а відносно сторони $x = x'$: (3.9). Результат перетворень приведений на рис. 3.14.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.6}$$

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.7}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -y' & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & y' & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 2y' & 1 \end{bmatrix}, \tag{3.8}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 2x' & 0 & 1 \end{bmatrix}. \tag{3.9}$$

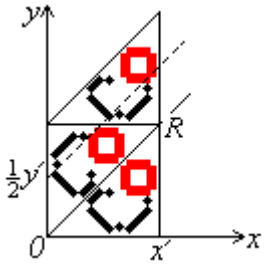


Рис. 3.13. Група *cm*, породжена відображенням відносно *y=x* і ковзним відображенням.

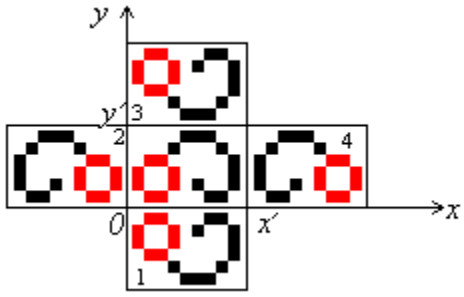


Рис. 3.14. Група *pmm*, породжена 4-ма відображеннями.

pmg - будується при $y'=x'$ і складається з відображення відносно *y=x* (R) і 2-ох поворотів на 180° відносно середин сусідніх сторін. Отже, результуючі матриці перетворень наступні: 1-а - відображення відносно x=y: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$;

2-а - зміщення точки T_1 $(1/2x'; 0)$ в початок координат і поворот на 180°:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{2}x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2}x' & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ x' & 0 & 1 \end{bmatrix};$$

3-я - зміщення точки T_2 $(x'; \frac{1}{2}y')$ в початок координат і поворот на 180°:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x' & -\frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & \frac{1}{2}y' & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 2x' & y' & 1 \end{bmatrix}.$$

Результат перетворень приведений на рис. 3.15.

pgg - складається з двох ковзних відображень. Осі цих відображень паралельні прямим OX, OY, крок рівний сторонам прямокутника. Матриці перетворень і результуюча наступні: 1-е відображення (перенос середини прямокутника в початок координат, ковзне відображення відносно OX):

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{2}x' & -\frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2}x' & \frac{1}{2}y' & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ x' & y' & 1 \end{bmatrix};$$

2-е відображення (перенос середини прямокутника в початок координат, ковзне відображення відносно OY):

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{2}x' & -\frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & y' & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2}x' & \frac{1}{2}y' & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & y' & 1 \end{bmatrix}.$$

Результат перетворень приведений на рис. 3.16.

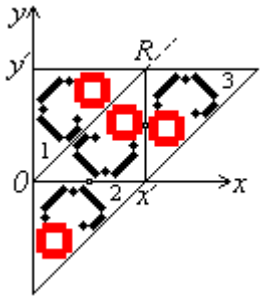


Рис. 3.15. Група *pmg*, породжена відображеннями і двома поворотами на 180°.

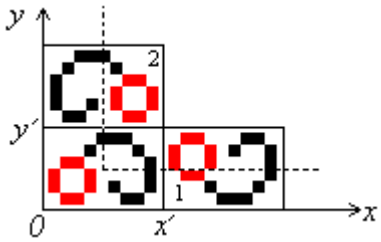


Рис. 3.16. Група *pgg*, породжена 2-ма перпендикулярними ковз. відображ.

cmm - будується при $y'=x'$ і складається з двох відображень відносно катетів трикутника OAB і повороту на 180° відносно середини гіпотенузи цього ж трикутника (рис.3.17). Матриці перетворень і їх результуючі наступні: 1-а (R_1)- відображення відносно x=y: (3.10); 2-а (R_2)- складається з переносу початку координат в вершину (0; y′) і відображення відносно y=-x: (3.11); 3-я - поворот відносно T(0; $\frac{1}{2}y'$): (3.12).

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad (3.10)$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -y' & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & y' & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ y' & y' & 1 \end{bmatrix}; \qquad (3.11)$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2}y' & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & y' & 1 \end{bmatrix}. \qquad (3.12)$$

Результат перетворень приведений на рис.3.17.

p4 - будується при $y'=x'$ і складається з повороту на 90° відносно перетину діагоналей квадрата і повороту на 180° відносно середини сторони. Матриці перетворень і їх результуючі мають вигляд: 1-а (перенос точки перетину діагоналей

S($\frac{1}{2}x';\frac{1}{2}y'$) в початок координат, поворот на 90°, перенос точки S в вихідне положення):

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{1}{2}x' & -\frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{2}x' & \frac{1}{2}y' & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ \frac{1}{2}x'+\frac{1}{2}y' & -\frac{1}{2}x'+\frac{1}{2}y' & 1 \end{bmatrix};$$

2-а - поворот відносно T(0; $\frac{1}{2}y'$):

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2}y' & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{2}y' & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & y' & 1 \end{bmatrix}.$$

Результат перетворень приведений на рис. 3.18.

p4m - будується при $y'=x'$ і складається з відображень відносно сторін трикутника OAB (рис.3.19). Перша матриця перетворень - відображення відносно гіпотенузи OB (чи відносно x=y): (3.13); 2-а -відносно катета OA (чи відносно осі OX): (3.14); 3-я - відносно катета AB (чи відносно $x=x'$): (3.15).

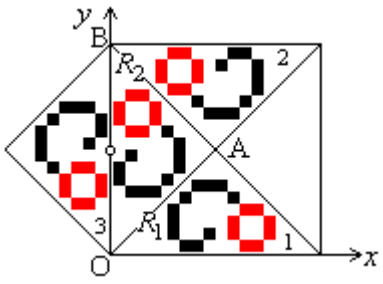


Рис 3.17. Група cmm, породжена двома відображеннями і поворотом на 180°.

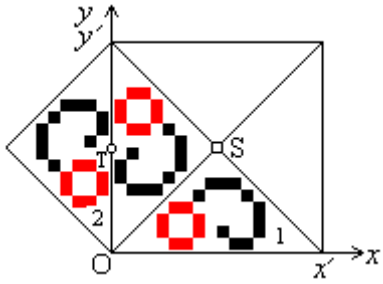


Рис. 3.18. Група p4, породжена поворотом на 90° і поворотом на 180°.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad (3.13)$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad (3.14)$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 2x' & 0 & 1 \end{bmatrix}. \qquad (3.15)$$

Результат перетворень приведений на рис. 3.19.

p4g - будується при $y'=x'$ і складається з відображення відносно $x=y$ (R, рис.3.20) і повороту на 90° відносно $S(x', 0)$. Матриці перетворень і їх результуючі наступні:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x' & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x' & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ x' & -x' & 1 \end{bmatrix}.$$

Результат перетворень приведений на рис. 3.20.

Отже, в даному підрозділі, на основі використання породжувальних перетворень формалізовано групи перетворень на площині та смузі, що дало можливість змоделювати узагальнений алгоритм.

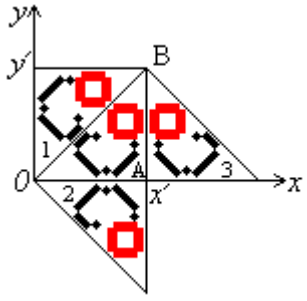


Рис. 3.19. Група p4m, породжена відображеннями в сторонах прямокутного рівнобедреного трикутника.

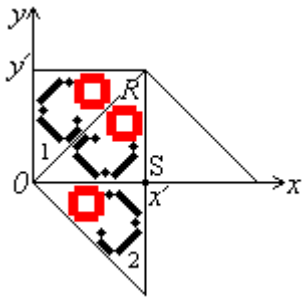


Рис.3.20. Група p4g, породжена відображенням і поворотом на 90°.

3.2. Алгоритми синтезу та моделювання мінімального рисунку

Символи мови запропонованої підрозділі 2.2 розіб'ємо на три групи. До першої (предметної групи) віднесемо непохідні елементи. Друга група - функціональні символи, що в нашому випадку відіграють роль символів операцій. До третьої віднесемо символи лівої і правої дужок. Будемо формувати сполучення за допомогою символів цих трьох груп, які назвемо термами [64]. Терми одиничної довжини - це предметні змінні, тобто непохідні елементи. Використовуючи метод математичної індукції, формалізуємо терми. Нехай для деякого $m>1$ терми довжиною, яка менша від m, визначені. Тоді терм - це слово довжиною m, якщо воно має вигляд $f(\alpha_1, \ldots, \alpha_n)$, де $\alpha_1, \ldots, \alpha_n$ - деякі терми меншої довжини, f - n-членний функціональний символ.

Задати значення предметного символу означає вказати непусту множину X і співставити з розглядуваним символом деякий елемент цієї множини. Це буде значенням даного символа. Співставити з функціональним символом одну з операцій, визначену на основній множині означає задати значення функціонального символа.

Значенням терма довжини 1 буде значення предметного символа, що складає цей терм. Для терму $f(\alpha_1, \ldots, \alpha_n)$ значенням буде деякий елемент x з X, який буде визначатися як значення операції f_0, що поставлена у відповідність символу f, в точці $x_1, \ldots, x_n$, де $x_1, \ldots, x_n$ - значення термів $\alpha_1, \ldots, \alpha_n$ [29].

Наприклад, термам a, $\sim b^2$, $a^3 \otimes \sim b^2$ відповідатимуть ланцюжки зображень (рис.3.21).

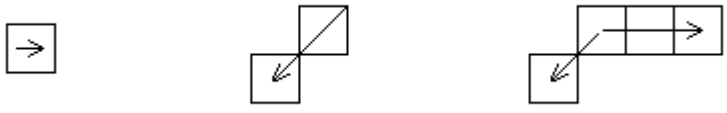


Рис. 3.21. Ланцюжки зображень.

З допомогою термів, маючи декілька операцій, які задані на певній множині X, можна задати нескінченну множину нових операцій, які будуть визначеними на ній же. Для цього фіксуємо операції і деякі предметні символи, а інші, задаючи довільно

в множині X, знаходимо значення терма.

Наприклад, значенням терма

$(\sim a^i) \oplus (\sim c^j)\ \Theta\ (\sim a^k \oplus b^l) \oplus (\sim c^m \otimes b^n)$, при $i = 4, j = 5, k = 4, l = 5, m = 5, n = 5$ буде зображення на рис. 3.22, а при $i = 1, j = 3, k = 1, l = 3, m = 3, n = 3$ - рис. 3.23.

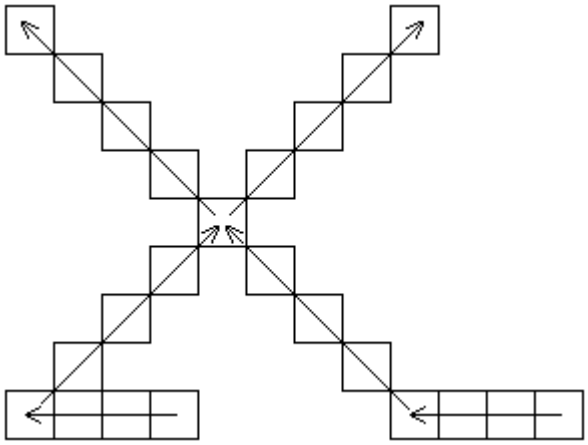


Рис. 3.22. Приклад 1-го терму.

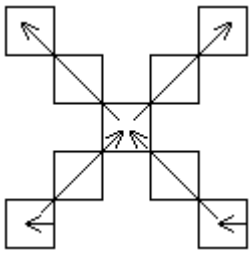


Рис. 3.23. Приклад 2-го терму.

В підрозділі 2.2 ми визначили алгебру, що складається з множини непохідних елементів E і множини визначених на E операцій F. Множина E називається основною множиною алгебри, а операції F називаються основними операціями. Виділимо в множині E якусь сукупність елементів &. Очевидно, що тоді для даних множин можна використати таку теорему [64] (теорема приводиться без доведення):

Теорема 2. Підмножина &*, яка породжена в E елементами сукупності & з допомогою операцій F, складається з елементів, що є значеннями термів, записаних з допомогою символів операцій F і символів деяких елементів &.

Таким чином з введенням поняття терма появилася можливість мінімальний рисунок записати у вигляді - Dmin= f($\alpha_1, \ldots, \alpha_n$), де $\alpha_1, \ldots, \alpha_n$ - деякі терми, f - n-членний функціональний символ. А звідси, орнаментне зображення на площині: z(Dmin) = z(f($\alpha_1, \ldots, \alpha_n$)), де $z \in Z$. Будь-яке зображення- орнамент на смузі: j(Dmin) = j(f($\alpha_1, \ldots, \alpha_n$)), де $j \in J$.

Як було показано в підрозділі 2.2 будь-який мінімальний рисунок може бути представлений за допомогою множини операцій F = {~, +, ⊕, -, Θ, *, ×, ⊗}. Набір операцій, з допомогою якого можна виразити будь-яке зображення назвемо повним базисом. Отже, множина F є повним базисом. Базис називається мінімальним, якщо

при виключенні хоча б однієї операції, яка входить в нього, набір операцій перетворюється в неповний [90, 91].

Покажемо, що операції множини F можна звести до операцій в базисах K={~, ⊕, ⊗}, L={~, ⊕, Θ}, M={~, ⊕}, виключень на непохідні елементи немає.

Демонструвати будемо графічно на елементах a^4, d^2, які зображені на рис.3.24. (елементи вибрані випадково, можна взяти будь-які інші).

Рис. 3.24. Вихідні елементи.

$a^4 + d^2$ (рис. 3.25) можна замінити виразом $a^4 \oplus c^2 \oplus d^2$ (рис. 3.26), в якому присутня лише операція ⊕ з множини M, яка є і в множинах K і L.

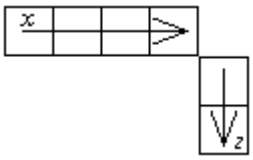


Рис. 3.25.Зображення $a^4 + d^2$.

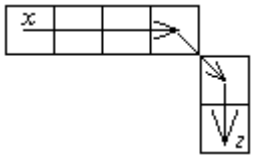


Рис. 3.26. Зображення $a^4 \oplus c^2 \oplus d^2$.

$a^4 \times d^2$ (рис. 3.27) можна замінити виразами: 1) $a^4 \otimes(\sim b^2) \oplus d^2$ (рис. 3.28);

2) $\sim((b^2) \ \Theta \ (\sim a^4)) \ \oplus \ d^2$ (використовуєм властивість 2.1 до виразу попереднього);

3) $a^4 \oplus \sim a^4 \oplus (\sim b^2) \oplus d^2$ (рис. 3.29).

В першому випадку операція × замінилася операціями з множини K, в другому - з множини L; в третьому - з множини M.

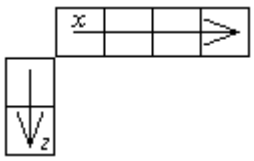


Рис. 3.27. Зображення $a^4 \times d^2$.

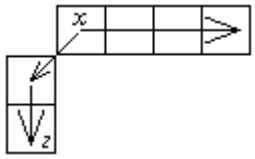


Рис.3.28. Зображення $a^4 \otimes(\sim b^2) \oplus d^2$.

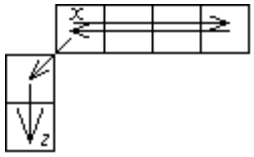


Рис. 3.29. Зображення $a^4 \oplus \sim a^4 \oplus (\sim b^2) \oplus d^2$.

$a^4 \otimes d^2$ (рис. 3.30) можна замінити виразами: 1) $a^4 \oplus {\sim}a^4 \oplus d^2$(рис. 3.31);

2) ${\sim}(({\sim} d^2) \Theta ({\sim}a^4))$.

Отже, одна з операцій множини K - $\otimes$ замінилася в першому випадку операціями множини M, в другому - множини L, застосувавши властивість 2.1 до вихідного виразу.

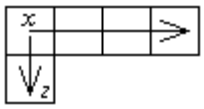


Рис. 3.30. Зображення $a^4 \otimes d^2$.

Рис. 3.31. Зображення $a^4 \oplus {\sim}a^4 \oplus d^2$.

a^4 - d^2 (рис. 3.32) можна замінити виразами: 1) $a^4 \oplus b^2 \Theta d^2$ (рис. 3.33);

2) $a^4 \oplus ({\sim}(({\sim}d^2) \otimes ({\sim}b^2)))$;

3) $a^4 \oplus b^2 \oplus {\sim}d^2 \oplus d^2$ (рис. 3.34).

Перший вираз містить операції множини L, другий - утворений застосуванням властивості 2.2 до попереднього виразу і містить операції множини K, третій - множини - M.

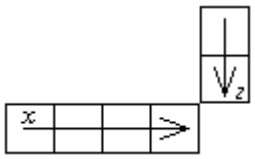


Рис. 3.32. Зображення a^4 - d^2.

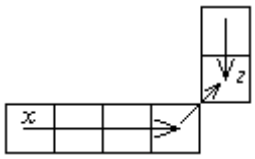


Рис. 3.33. Зображення $a^4 \oplus b^2 \Theta d^2$.

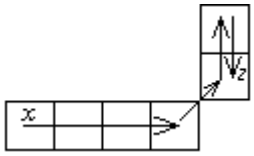


Рис. 3.34. Зображення $a^4 \oplus b^2 \oplus {\sim}d^2 \oplus d^2$.

$a^4 \Theta d^2$ (рис. 3.35) можна замінити виразами 1) $a^4 \oplus {\sim}d^2 \oplus d^2$ (рис. 3.36);

2) ${\sim}(({\sim}d^2) \otimes ({\sim}a^4))$ (згідно властивості 2.2).

Отже, операція множини L - Θ замінена в першому виразі операціями множини M, в другому - множини K, застосувавши властивість 2.1 до вихідного зображення.

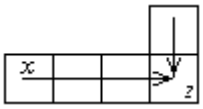


Рис. 3.35. Зображення $a^4 \Theta d^2$.

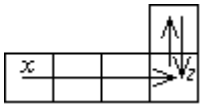


Рис. 3.36. Зображення $a^4 \oplus {\sim}d^2 \oplus d^2$.

$a^4 * a^4$ (рис. 3.37) можна замінити виразами 1) $a^4 \otimes d^2 \oplus a^4$ (рис. 3.38)

2) $\sim((\sim d^2) \Theta (\sim a^4)) \oplus a^4$;

3) $a^4 \oplus d^2 \oplus (\sim a^4) \oplus a^4$ (рис. 3.39).

Операція $*$ замінена операціями множини K, в першому виразі, операціями множини L (застосувавши властивість 2.2 до попереднього виразу) - в другому виразі, операціями множини M - в третьому виразі.

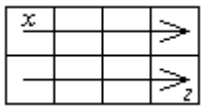


Рис. 3.37. Зображення $a^4 * a^4$.

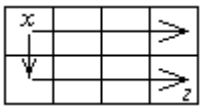


Рис.3.38. Зображення $a^4 \otimes d^2 \oplus a^4$.

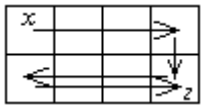


Рис. 3.39. Зображення $a^4 \oplus d^2 \oplus (\sim a^4) \oplus a^4$.

Таким чином множини операцій K={~, ⊕, ⊗}, L={~, ⊕, Θ} є повними базисами, M={~, ⊕} є мінімальним базисом. Повним базисом буде також U ={~, ⊕, *}, якщо в зображенні зустрічаються ланцюжки однакової довжини.

Досить ясне представлення про множини операцій дає наступна теорема:

Теорема 3. Будь-яке зображення можна представити з допомогою множини непохідних елементів та набору операцій M={~, ⊕}. Більше того ці операції є незалежні.

Доведення.

Наступні еквівалентні вирази демонструють те, що будь-яка операція з набору {+, -, Θ, *, ×, ⊗} *замінюється операціями* {~, ⊕}.

$$U_1 \otimes U_2 = U_1 \oplus {\sim}U_1 \oplus U_2;$$

$$U_1 \Theta U_2 = U_1 \oplus {\sim}U_2 \oplus U_2;$$

$$U_1 \times U_2 = U_1 \oplus {\sim}U_1 \oplus x^2 \oplus U_2;$$

$$U_1 - U_2 = U_1 \oplus x^2 \oplus {\sim}U_2 \oplus U_2;$$

$$U_1 * U_2 = U_1 \oplus x^2 \oplus {\sim}U_2 \oplus U_2;$$

$$U_1 + U_2 = U_1 \oplus x^2 \oplus U_2,$$

де x^2*- непохідний елемент довжиною два, який залежить від напрямку* U_2; $\oplus$ *є незалежна операція від* ~, *бо вона є операцією конкатенації.* ~ *є унарною операцією і незалежною від* $\oplus$, *тому що вирази виду* $(a \oplus (\sim b))$ *ніяким чином не заміняться операцією* $\oplus$.

Множина { +, -, Θ, *, ×, ⊗} хоча і не є необхідною все ж дуже зручною, особливо для опису зображень з простою структурою. Покажемо приклад побудови мінімального рисунку в декількох базисах.

В базисі A={~, ⊕, ⊗} (рис. 3.40):

$b^2 \oplus (\sim d^3 \otimes a^3) \oplus nd^3 \oplus \sim a^3 \oplus d^3 \oplus na^3 \oplus a^2 \oplus b^5 \oplus (\sim c^3 \otimes \sim d^3) \oplus nd^3 \oplus (\sim a^3 \otimes d^3)$

В базисі B={~, ⊕, Θ} (на основі властивості 2.1):

$b^2 \oplus (\sim(\sim a^3 \Theta d^3)) \oplus nd^3 \oplus \sim a^3 \oplus d^3 \oplus na^3 \oplus a^2 \oplus b^5 \oplus (\sim(d^3 \Theta c^3)) \oplus nd^3 \oplus (\sim(\sim d^3 \Theta a^3))$.

В базисі C={~, ⊕} (рис. 3.41):

$b^2 \oplus (\sim d^3) \oplus nc^3 \oplus (\sim a^3) \oplus nd^3 \oplus a^3 \oplus (\sim nb^3) \oplus (\sim d^3) \oplus nc^3 \oplus a^2 \oplus b^4 \oplus (\sim c^2) \oplus a^2 \oplus (\sim d^2) \oplus (\sim c^2) \oplus na^3 \oplus d^5$.

В базисі={~, ⊕, ⊗, Θ} (на основі властивостей 2.1, 2.2):

$b^2 \oplus (\sim d^3 \otimes a^3) \oplus nd^3 \oplus (\sim a^3 \Theta \sim d^3) \oplus nc^3 \oplus a^2 \oplus b^5 \oplus (\sim c^3 \otimes \sim d^3) \oplus nd^3 \oplus (\sim a^3 \otimes d^3)$.

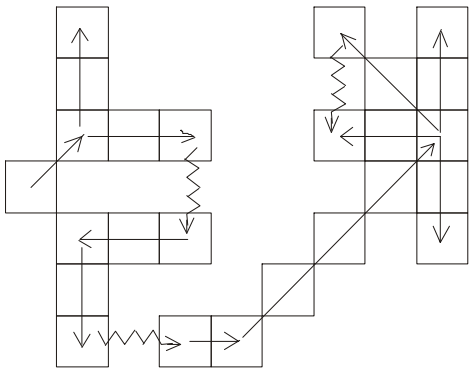


Рис.3.40.Приклад генерування в базисі A.

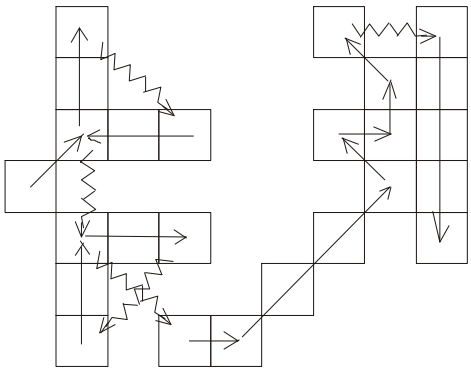


Рис.3.41. Приклад генерування в базисі C.

Коли необхідно задати не чорно-білий мінімальний рисунок, а кольоровий, тоді вводимо ще один індекс до непохідного елемента (крім довжини) - індекс кольору. Він приводиться біля непохідного елемента в круглих дужках так як показано на рис. 3.42:

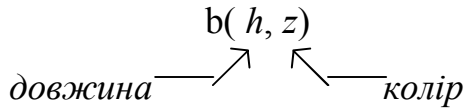


Рис. 3.42. Кодування кольору та довжини елемента.

Коди кольорів і їх значення приводяться в додатку З на рис. З.12.

Вишивка рідко зустрічається у вигляді контурів [43-45]. Найчастіше вишивальниці будують складні і прості геометричні фігури заповнені одним і тим же кольором (області). Для їх опису достатньо [29] описати контури фігури и ввести позначку @(*x*, *y*, *z*), де *x*, *y*, - координати будь-якої точки, що міститься всередині фігури, *z* - код кольору. При наявності даного позначення редактор орнаментів (див. підрозділ 4.1) автоматично заповнює область, яка містить точку (*x*,*y*) кольором *z*.

В підрозділі 1.2 зроблено групування мінімальних рисунків на наступні типи:

- геометричні;
- рослинні;
- рослинно- геометричні;
- зооморфні.

Приведемо приклади рисунків кожного типу, реалізованих в базисі C={~, ⊕} з допомогою редактора орнаментів (див. підрозділ 4.1). В додатку **В** показано їх геометричне представлення, а також і інших мінімальних рисунків (формульне представлення знаходиться в додатку Д).

Для мінімального рисунку рис. В.2.б. буде наступна формула:

d(2,1) + b(2,1) + c(2,1) + b(4,1) + a(4,1) + (~d(2,1))+ b(2,1) + c(2,1) + (~a(2,1))+ d(2,1) + (~b(3,1))+ (~a(4,1))+ @(7,3,1) + (~b(3,0))+ (~a(2,0))+ d(3,1) + c(2,1) + d(4,1) + +c(3,1) +(~d(6,1))+(~c(3,1))+ d(2,1) + c(2,1) + @(5,10,1) + (~d(4,0))+ a(2,1) + d(2,1) + +c(2,1)+(~d(2,1))+ a(3,1) + c(5,1) + d(4,1) + (~c(2,1))+ (~a(2,1))+ (~c(4,1))+ (~d(2,1))+

+@(8,8,1) + c(7,0) + (~d(2,0))+ (~d(2,1))+ c(2,1) + a(7,1) + b(2,1) + a(3,1) + b(2,1) + +a(2,1) + b(2,1) + a(2,1) + (~a(14,0)) + d(2,0) + a(6,1)+ (2,1)+a(3,1)+b(5,1)+(~d(4,1))+ +(~c(3,1))+ (~a(5,1))+ d(2,1) + (~b(2,1))+ d(2,1) + a(4,1) + c(2,1) + (~b(2,1))+ +(~d(2,1))+ (~a(4,1))+ d(5,1) + a(2,1) + (~d(2,1))+ (~a(3,1))+ b(2,1) + b(2,1) + a(2,1) + +(~d(4,0))+(~a(4,0))+(~d(4,1))+b(2,1) + a(2,1) + (~b(2,1))+ d(3,1) + (~a(6,1))+ b(2,1) + +(~d(2,1))+ c(2,1) + d(3,0) + a(2,14) + c(2,14) + d(4,14) + (~a(2,14))+ (~b(2,14))+ +(~a(3,14))+ (~c(3,14))+ (~d(3,14))+ b(2,14) + a(3,14) + @(16,5,14).

Для рисунку рис. В.2.д. формула має вигляд:

(~c(2,1))+ na(4,1) + d(2,10) + (~a(2,10))+ nd(3,10) + d(3,10) + b(2,10) + d(2,10) + +d(2,1) + (~a(3,1))+ (~d(4,10))+ (~b(2,10))+ d(2,10) + (~b(2,10))+ (~a(2,1))+ +(~c(2,1))+ (~d(2,1))+ b(2,1) + d(2,1).

Для рисунку рис. В.1.ж.буде наступна формула:

a(2,10) + (~b(3,1))+ c(2,1) + d(3,10) + c(5,10) + a(3,10) + b(3,10) + (~d(2,10))+ +(~c(2,10))+ d(3,10) + (~a(6,10))+ (~d(6,10))+ a(3,10) + (~c(2,10))+ (~a(2,10))+ +(~b(2,10))+ nd(8,0) + (~c(2,10))+ d(2,10) + (~b(2,10))+ (~nd(5,0))+ c(5,1) + +(~nd(6,0))+ na(2,0) + (~d(2,1))+ c(2,1) + (~d(2,1))+ b(2,1) + (~d(4,1))+ (~c(3,1))+ +(~a(4,1))+(~b(2,1))+a(5,1)+ c(6,1) + (~d(2,1))+ (~c(3,1))+ d(3,1) + nd(8,0) + na(2,0) + +b(2,10) + nd(2,0) + a(1,1) + (~nd(11,0))+ (~c(4,1))+ (~a(7,10))+ @(1,0,10) + nd(9,0) + +na(4,0) + (~d(2,16)) .

Вигляд формули для рис. В.3.а. наступний:

(~d(3,1))+ b(3,1) + a(3,1) + c(4,1) + b(2,1) + (~d(2,1))+ c(2,1) + (~a(2,1))+ d(3,0) + +c(6,1) + a(2,1) + b(6,1) + d(3,1) + (~b(5,1))+ c(5,1) + d(3,1) + (~c(3,1))+ (~a(3,1))+ +d(3,1) + (~a(2,1))+ (~d(3,0))+ (~a(7,1))+ d(3,1) + (~a(2,1))+ (~d(3,0))+ (~a(6,1))+ +(~c(3,1))+(~d(3,1))+b(4,1) + (~d(3,1))+ (~c(2,1))+ (~a(2,1))+ a(8,0) + c(3,1) + b(8,1) + +a(2,1) + d(2,1) + (~b(7,1))+ (~a(2,0))+ b(7,12) + a(2,0) + d(7,0) + (~b(8,10))+ +(~d(3,10))+(~a(7,11))+(~d(3,11))+a(8,11)+(~c(2,11))+ (~a(9,13))+ b(2,13) + a(7,13) + +(~c(2,13))+ (~a(6,13))+ @(5,4,10) + d(4,0) + d(3,10) + (~a(2,10))+ (~d(3,10))+ +@(2,13,12) + @(8,11,11) + @(15,13,10) + @(19,15,12).

Примітка. В формулах операцію + слід читати як ⊕ (див. підрозділ 2.2).

Проведений синтез мінімальних рисунків, які були виділені в предметній області (українська народна вишивка) показує, що їх необхідно відповідним чином упорядкувати. Для цього використаємо інтегроване сховище даних, яке забезпечує незалежність даних від прикладних програм, тобто базу даних (БД) [92-96]. Як відомо, проектування баз даних - це ітераційний та багатоетапнний процес під час аналізу інформаційної моделі предметної області. При проектуванні будемо використовувати дворівневе [92] подання даних: інфологічне та логічне (даталогічне). Інфологічний рівень - це інформаційно-логічна модель предметної області. В даній предметній області виділені наступні об’єкти: орнамент, підорнамент, мінімальний рисунок. Інфологічна модель представлена на рис. 3.43. Для графічного представлення інфологічної моделі використані діаграми Чена. Наступний рівень проектування - даталогічний. На цьому рівні формується концептуальна модель даних. Даталогічна модель представлена на рис. 3.44.

Таким чином, в даному підрозділі на основі запропонованої мови опису, розроблені алгоритми синтезу мінімального рисунку: розглянуто моделювання мінімального рисунку в різних функціональних базисах, доведена теорема про мінімальний базис, показані приклади їх застосування до опису мінімальних зображень та спроектована база даних.

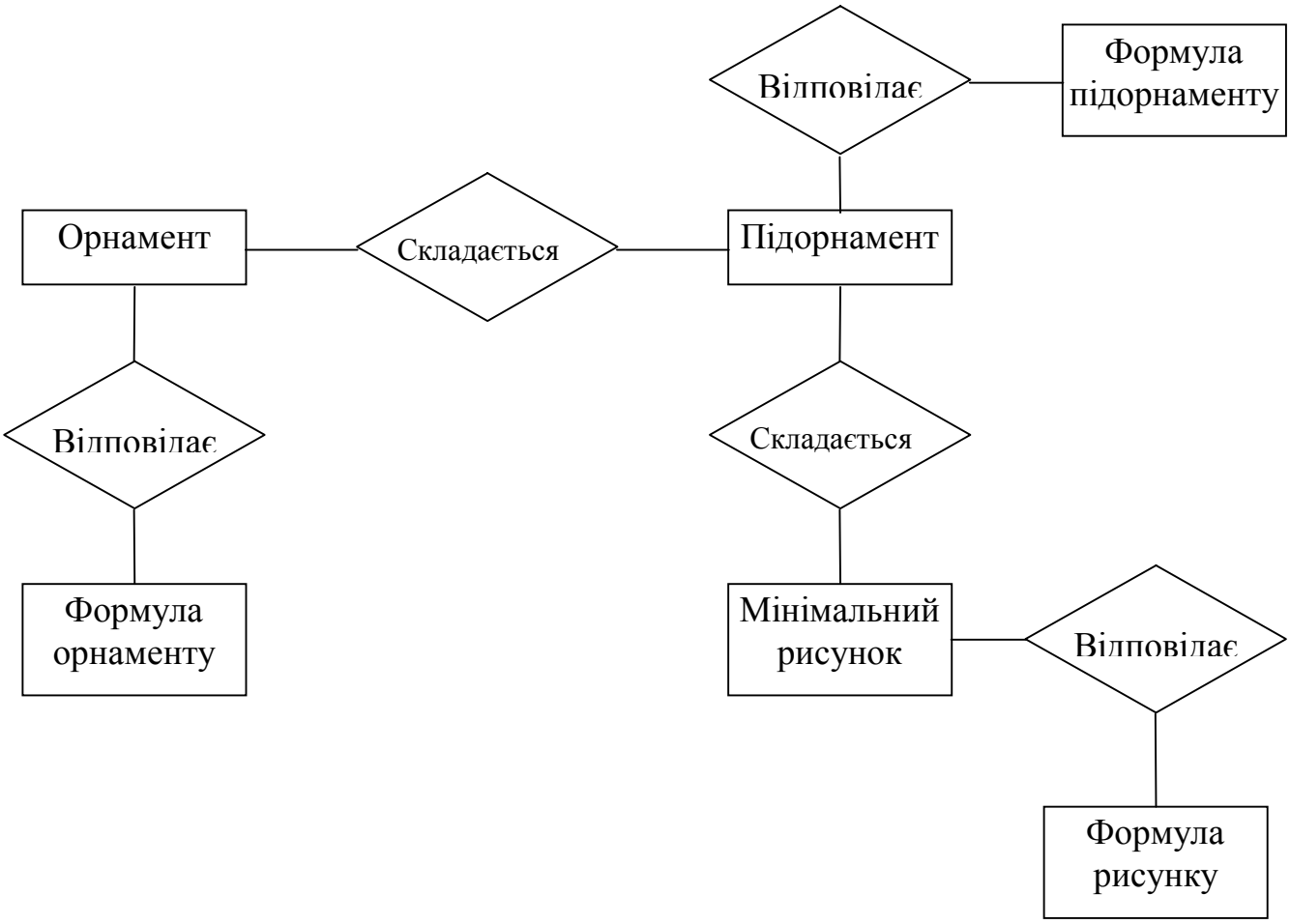


Рис. 3.43. Інфологічна модель.

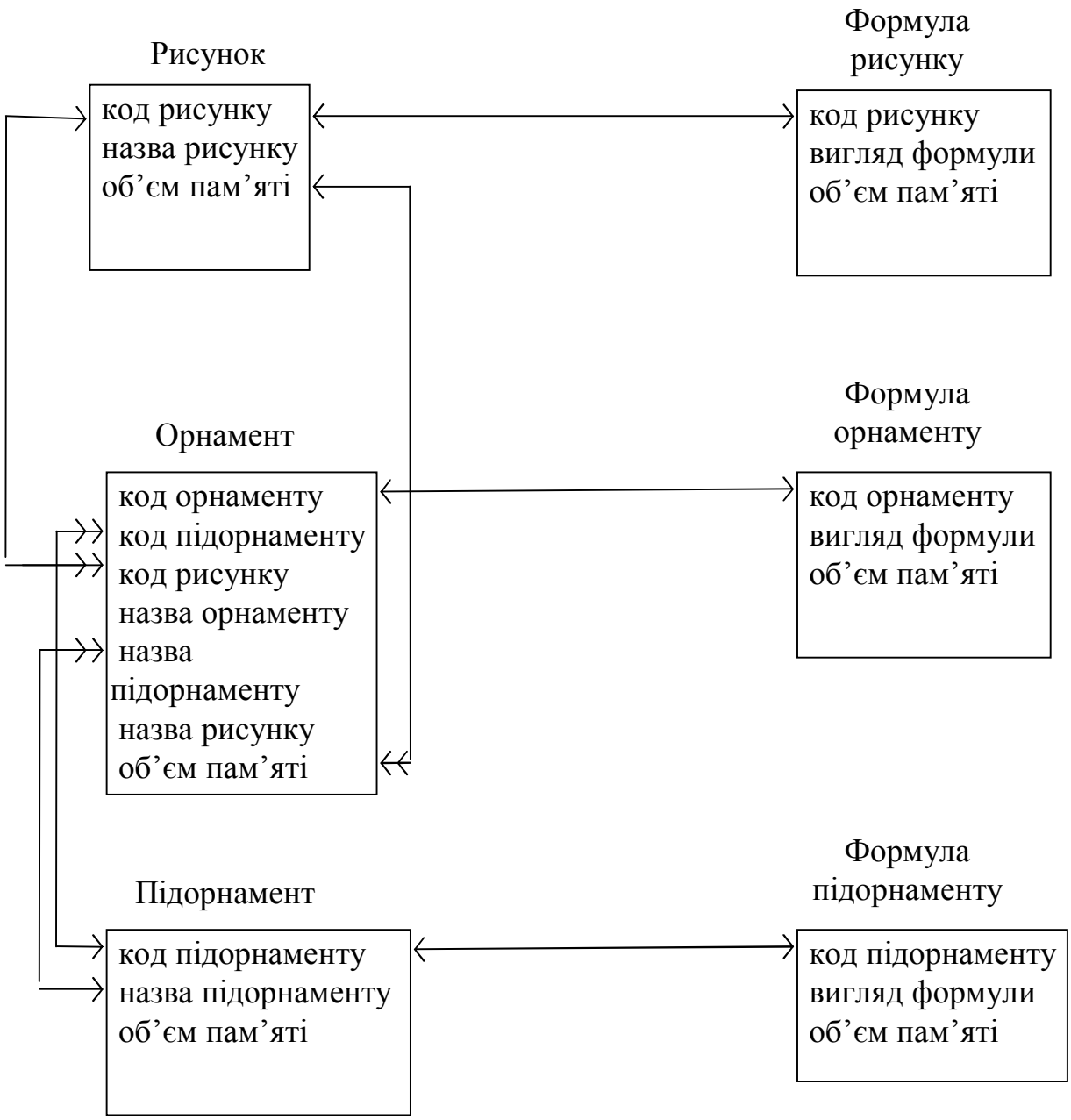


Рис. 3.44. Даталогічна модель.

3.3. Синтез, моделювання та архівування складних зображень-орнаментів

3.3.1. Узагальнений алгоритм синтезу та моделювання зображень-орнаментів

Запропоновані алгоритми побудови груп перетворень (підрозділ 3.1), мінімального рисунку (підрозділ 3.2) дають змогу розробити узагальнений алгоритм синтезу складних зображень-орнаментів.

Як відомо, синтез це створення зображень в автроматизованому (з участю людини) або в автоматичному режимах та відображення його з метою візуального спостереження [97]. Синтез зображень-орнаментів складається з наступних етапів [29,30]: 1) формування мінімального рисунку; 2) формування груп перетворень зображень.

Орнамент - строге композиційне розміщення мінімальних рисунків (підрозділ 1.2, 2.1). Отже, для синтезу необхідно спочатку сформувати мінімальний рисунок. Для його побудови (підрозділ 2.2) використовуємо мову опису, яка представляє собою множину непохідних елементів з алгебраїчними правилами їх поєднання. Синтез мінімального рисунку відбувається в одному випадку на основі його алгебраїчної формули опису, в іншому, шляхом використання бази типових для даної орнаментальної області мінімальних рисунків. При цьому можливі два способи формування мінімального рисунку: детермінований і стохастичний [29]. Використання детермінованого способу синтезу вимагає повного формалізованого опису мінімального рисунку. Стохастичний спосіб дозволяє шляхом використання генератора випадкових чисел із різними законами розподілу дискретних випадкових величин вибирати із відповідних множин (множина непохідних елементів, множина кольорів, множина довжин непохідних елементів, множина операцій над непохідними елементами) необхідний елемент з довжиною та кольором. При виборі унарної операції проходить автоматичне її виконання над сформованим ланцюжком елементів. В противному разі (бінарні операції) відбувається вибір другого ланцюжка елементів.

Наступним етапом синтезу орнаменту є формування груп перетворень зображень (формування підорнаменту), які можуть бути представлені на смузі та на

площині. Формування може здійснюватися трьома способами:

1) через породжуючі перетворення - переноси, відображення, повороти, ковзні відображення (підрозділ 2.1) [88];

2) на основі осьових симетрій (підрозділ 3.1) [29, 30];

3) рекурсивним способом (як підгрупи групи переносів) (підрозділ 2.3) [31].

Після синтезу підорнаменту (підорнамент є складовою орнаменту) формуємо орнамент. Утворення орнаменту із підорнаментів відбувається шляхом послідовного заповнення площини групами перетворень зображень, причому послідовність заповнення немає принципового значення і відбувається за рішенням людини-орнаментиста.

Можливі комбінації синтезу складових орнаменту (мінімального рисунку та груп перетворень) приведені в табл. 3. 1 .

Таблиця 3.1

Комбінації синтезу складових орнаменту

Складові орнаменту	Синтез	
	Детермінований	Стохастичний
Мінімальний рисунок	Формула опису (апріорна)	Формула опису (автоматично сформована)
Групи перетворень	• через породжуючі перетворення • на основі осьових симетрій • рекурсивним способом	—

Розглянемо детермінований спосіб синтезу зображень. Із запропонованої мови опису мінімального рисунку (підрозділ 2.2) випливає, що маючи множину непохідних елементів E та множину операції над ними F можна синтезувати довільний чорно-білий мінімальний рисунок. Для синтезу кольорового мінімального рисунку вводимо множину кольорів непохідних елементів. Слід замітити, види непохідних елементів та послідовність операцій над ними та їх кольори немає принципового значення, тобто вони можуть бути довільними. Але при цьому

необхідно дотримуватись обмеження на площу, яку повинен займати мінімальний рисунок.

Нехай необхідно синтезувати мінімальний рисунок, який заданий наступною формулою (термом):

$$(\sim a^i) \oplus (\sim c^j)\ \Theta\ (\sim a^k \oplus b^l) + (c^m \otimes d^n),$$

де i, j, k, l, m, n - довжини відповідних елементів.

На рис. 3.45 представлена структурна схема, яка відповідає даному виразу.

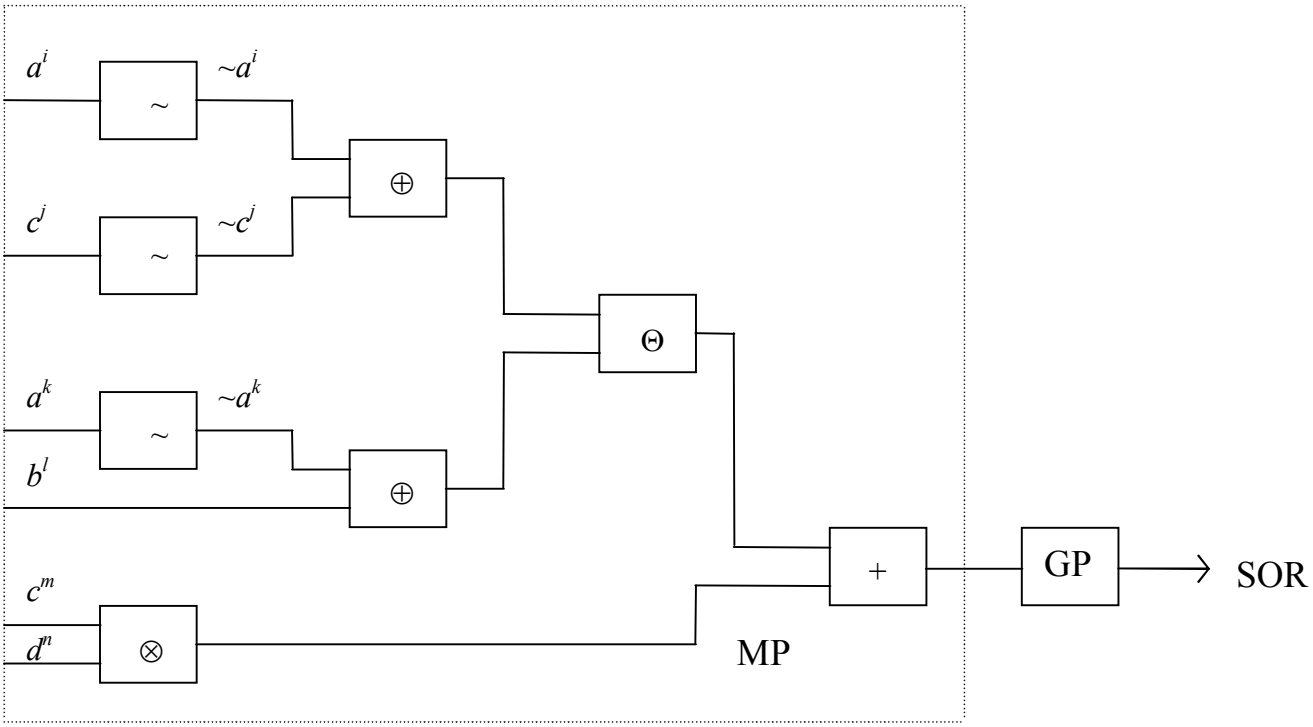


Рис. 3.45. Структурна схема терму $(\sim a^i) \oplus (\sim c^j)\ \Theta\ (\sim a^k \oplus b^l) + (c^m \otimes d^n)$.

На рис. 3.45 приведені наступні позначення:

MP -мінімальний рисунок;

GP - група перетворень зображень;

SOR - підорнамент.

Після синтезу мінімального рисунку за допомогою відповідних груп перетворень (алгоритми синтезу груп перетворень описані в підрозділі 3.1) отримуємо підорнамент - SOR.

Як було показано вище орнамент складається з підорнаментів, тобто

$$SOR_1 \cup SOR_2 \cup ... \cup SOR_s = OR.$$

Таким чином, маючи множину підорнаментів, які в свою чергу складаються із

мінімального рисунку та групи перетворень, шляхом послідовного їх розгортання формуємо орнамент. Процес синтезу орнаменту графічно представлений на рис. 3.46.

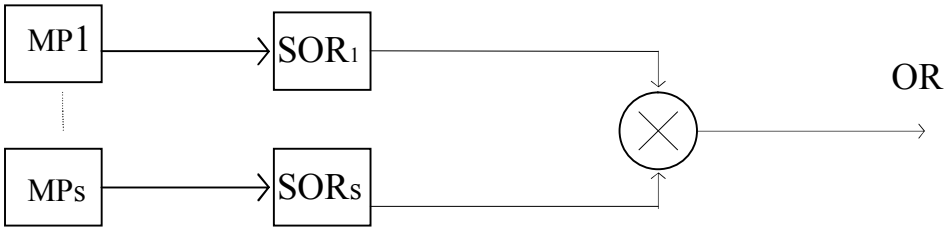


Рис. 3.46. Процес синтезу орнаменту.

Приведемо приклад синтезу мінімального рисунку і групи перетворень. На рис. 3.47 мінімальний рисунок розкладено на непохідні елементи і показана конкатенація між ними. Рис. 3.48 демонструє мінімальний рисунок згенерований по формулі: $(c^2 \oplus (\sim d^3) + (\sim nd^3) + (\sim d^2) \oplus (\sim b^7) \oplus c^4 + a^2)\ \Theta\ ((\sim b^5) \oplus c^5) \oplus ((\sim b^2) + (\sim a) + (\sim na^2) \oplus (\sim c^4) + nd^2 + d)$.

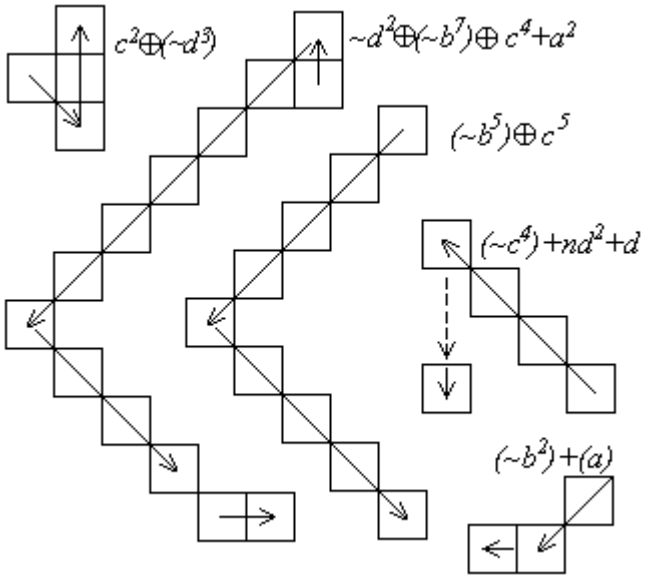


Рис. 3.47. Непохідні елементи і конкатенація між ними.

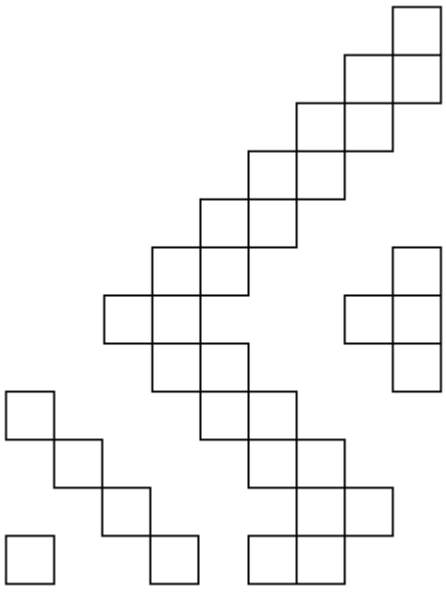


Рис. 3.48. Зображення, утворене з елементів рис. 3.47.

На рис. 3.49 представлено орнамент, що складається з одного підорнаменту і реалізований по формулі :

p1m /10/: $(c^2 \oplus (\sim d^3) + (\sim nd^3) + (\sim d^2) \oplus (\sim b^7) \oplus c^4 + a^2)\ \Theta\ ((\sim b^5) \oplus c^5) \oplus ((\sim b^2) + (\sim a) +$

$(\sim na^2) \oplus (\sim c^4) + nd^2 + d)$, де 10 - число ітерацій (кількість мінімальних рисунків, що присутнє в орнаменті).

Рис. 3.49. Зображення-орнамент.

Стохастичний спосіб синтезу зображень базується на алгоритмах генерування випадкових величин [98-100]. Для імітації випадкових подій використаємо схему випробувань «за жеребкуванням» [98]. Припустимо, що в результаті спроби настає одна з n несумісних подій A_1, A_2, ..., A_n, які утворюють повну групу подій із ймовірностями p_i, $(i=\overline{1,n})$ для яких справедлива тотожність $\sum_{i=1}^{n} p_i = 1$. Тоді алгоритм випробувань «за жеребкуванням» представляється наступним чином:

1) проводимо розбиття відрізка [0,1] на n частин довжиною $p_1, p_2,..., p_i,..., p_n$. Таким чином точки поділу відрізка матимуть координати: $l_0=0$, $l_1= p_1$, $l_2= p_1 + p_2,...$, $l_n=\sum_{i=1}^{n} p_i = 1$;

2) проводимо вибір рівномірно розподіленого випадкового числа ξ, використовуючи генератор випадкових чисел на відрізку [0,1]. При виконанні нерівності $l_{k-1} \le \xi < l_k$ будемо вважати, що відбулася подія A_k, тому що $P\{A_k\} = P\{l_{k-1} \le \xi < l_k\} = l_k - l_{k-1} = p_k$.

Для нашого випадку існують наступні множини подій: A - множина подій вибору непохідних елементів, B - множина подій вибору кольорів непохідних елементів, C - множина подій вибору довжин непохідних елементів, D - множина подій вибору операцій над непохідними елементами.

Множина A є наступною: $A=\{A_1, ..., A_8\}$, де A_1 - подія - вибрати непохідний елемент a, A_2 - подія - вибрати непохідний елемент b, A_3 -подія - вибрати непохідний елемент c, A_4 - подія - вибрати непохідний елемент d, A_5 - подія - вибрати непохідний

елемент *на,* A_6 - подія - вибрати непохідний елемент *нb,* A_7 - подія - вибрати непохідний елемент *нс,* A_8 - подія - вибрати непохідний елемент *нd* (див. підрозділ 2.2). Порожні елементи виключаємо, тому що множина кольорів (див. далі) містить колір фону - білий. Всі події A_1, ..., A_8 рівноймовірні, тобто $P(A_1)= p_1=1/8$, ..., $P(A_8)= p_8=1/8$. Сукупність подій A_1, ..., A_8 утворює повну групу подій.

Множина $B = \{B_1,...,B_{16}\}$, де B_1 - подія - вибрати непохідний елемент чорного кольору, B_2 - подія - вибрати непохідний елемент коричневого кольору, B_3 - подія - вибрати непохідний елемент зеленого кольору, B_4 - подія - вибрати непохідний елемент оранжевого кольору, B_5 - подія - вибрати непохідний елемент фіолетового кольору, B_6 - подія - вибрати непохідний елемент вишневого кольору, B_7 - подія - вибрати непохідний елемент темнозеленого кольору, B_8 - подія - вибрати непохідний елемент темносірого кольору, B_9 - подія - вибрати непохідний елемент сірого кольору, B_{10} - подія - вибрати непохідний елемент червоного кольору, B_{11} - подія - вибрати непохідний елемент світлозеленого кольору, B_{12} - подія - вибрати непохідний елемент жовтого кольору, B_{13} - подія - вибрати непохідний елемент синього кольору, B_{14} - подія - вибрати непохідний елемент рожевого кольору, B_{15} - подія - вибрати непохідний елемент голубого кольору, B_{16} - подія - вибрати непохідний елемент білого кольору. Всі події B_1, ..., B_{16} розподілені по деяким законам з імовірностями $q_1, ..., q_{16}$. Сукупність подій B_1, ..., B_{16} утворює повну групу подій.

Множина $C = \{C_1, ..., C_{50}\}$ - події вибору значень довжин ланцюжків елементів, які є цілими числами від 1 до 50. Всі події C_1, ..., C_{50} розподілені по деяким законам з імовірностями $t_1, ..., t_{50}$. Сукупність подій C_1, ..., C_{50} утворює повну групу подій.

Множина D включатиме наступні події вибору операцій $D = \{D_1, ..., D_4\}$, де D_1 - подія - вибрати операцію ~, D_2 - подія - вибрати операцію $\oplus$, D_3 - подія - вибрати операцію Θ, D_4 - подія - вибрати операцію $\otimes$. Операція * не включена в множину D, тому що вона вимагає однакової довжини ланцюжків елементів, що випадковим чином відбувається не часто. Всі події D_1, ..., D_4 рівноймовірні, тобто $P(D_1)=1/4$, ..., $P(D_4)=1/4$. Сукупність подій D_1, ..., D_4 утворює повну групу подій.

Формуємо повну групу подій: $R_1=A_1B_1C_1$ - подія полягає у виборі непохідного елемента *а* чорного кольору одиничної довжини; $R_2=A_1B_1C_2$ - подія полягає у виборі непохідного елемента *а* чорного кольору довжиною 2; . . . ; $R_{50}=A_1B_1C_{50}$ - подія полягає у виборі непохідного елемента *а* чорного кольору довжиною 50; $R_{51}=A_1B_2C_1$ - подія полягає у виборі непохідного елемента *а* коричневого кольору одиничної довжини; . . . ; $R_{6400}=A_8B_{16}C_{50}$ - подія полягає у виборі непохідного елемента *нd* білого кольору довжиною 50. Загальна кількість подій буде рівна 8×16×50 = 6400.

Обчислюємо ймовірності цих подій за правилом: імовірність добутку незалежних подій дорівнює добутку їх ймовірностей [101, 102]. Маємо: $P(R_1) = P(A_1B_1C_1) = p_1 \cdot q_1 \cdot t_1$, $P(R_2) = P(A_1B_1C_2) = p_1 \cdot q_1 \cdot t_2$, ..., $P(R_{6400}) = P(A_8B_{16}C_{50}) = p_8 \cdot q_{16} \cdot t_{50}$.

Поділяємо відрізок [0,1] на 6400 частин. Координати точок поділу: l_0=0, l_1= $p_1 \cdot q_1 \cdot t_1$, l_2= $p_1 \cdot q_1 \cdot t_1 + p_1 \cdot q_1 \cdot t_2$,..., l_n=1.

Вибираємо наступне випадкове число ξ . Якщо $l_{k-1} \le \xi < l_k$, то імітується подія R_k.

Розглянемо інший підхід. Нехай Z - випадкова величина - вибір відповідного непохідного елемента. Тоді $A_1 = \{Z = z_1\}$, $A_2 = \{Z = z_2\}$, ..., $A_8 = \{Z = z_8\}$ і Z приймає значення 1, ..., 8. Ряд розподілу дискретної випадкової величини має вигляд:

Можливі значення	z_1	z_2	...	z_8
Імовірності	p_1=1\8	p_2=1\8	...	p_8=1\8

Нехай X - випадкова величина - вибір відповідного кольору. Тоді $B_1 = \{X = x_1\}$, $B_2 = \{X = x_2\}$,..., $B_{16} = \{X = x_{16}\}$ і X приймає значення 1, ..., 16. Ряд розподілу дискретної випадкової величини має вигляд:

Можливі значення	x_1	x_2	...	x_{16}
Імовірності	q_1	q_2	...	q_{16}

Нехай Y - випадкова величина - вибір довжини ланцюжка елементів. Тоді $C_1 = \{Y = y_1\}$, $C_2 = \{Y = y_2\}$, ..., $C_{50} = \{Y = y_{50}\}$ і Y приймає значення 1, ..., 50. Ряд розподілу

дискретної випадкової величини має вигляд:

Можливі значення	y_1	y_2	...	y_{50}
Імовірності	t_1	t_2	...	t_{50}

Для випадку дискретних випадкових величин X і Y імовірності q_k і t_k пов'язані рекурентним співвідношенням

$$p_{k+1} = p_k r(k),$$

де $r(k)$ - деяка функція, що залежить від значення індексу k.

Для біноміального розподілу (розподілу Бернуллі) з параметрами p і n

$$p_k = P(X = k) = C_n^k p_k (1-p)^{n-k}.$$

Тоді

$$r(k) = \frac{p_{k+1}}{p_k} = \frac{n!}{(k+1)!(n-k-1)!} \frac{k!(n-k)!}{n!} \frac{p}{1-p} = \frac{n-k}{k+1} \frac{p}{1-p}.$$

Для розподілу Пуассона з параметром λ

$$p_k = \frac{\lambda^k e^{-\lambda}}{k!}, \quad r(k) = \frac{\lambda^{k+1} e^{-\lambda}}{(k+1)!} \frac{k!}{\lambda^k e^{-\lambda}} = \frac{\lambda}{k+1}.$$

Для геометричного розподілу з параметром p

$$p_k = p(1-p)^k, \ \ r(k) = 1-p.$$

На основі запропонованих способів синтезу орнаментів розробимо узагальнений алгоритм їх синтезу (рис. 3.50).

1. Вводимо характеристики орнаменту :

$k_з$ - коефіцієнт заповнення площі синтезу орнаменту;

$m_{мр}$ - кількість мінімальних рисунків в рапорті;

$m_{по}$ - кількість підорнаментів.

2. На основі запропонованої мови опису складних зображень (підрозділ 2.1) та алгоритмів синтезу мінімального рисунку (підрозділ 3.2) синтезуємо мінімальний рисунок.

3. Використовуючи другий крок алгоритму, синтезуємо групу перетворень (підорнамент) на базі запропонованих алгоритмів синтезу груп (підрозділ 2.3, 3.1).

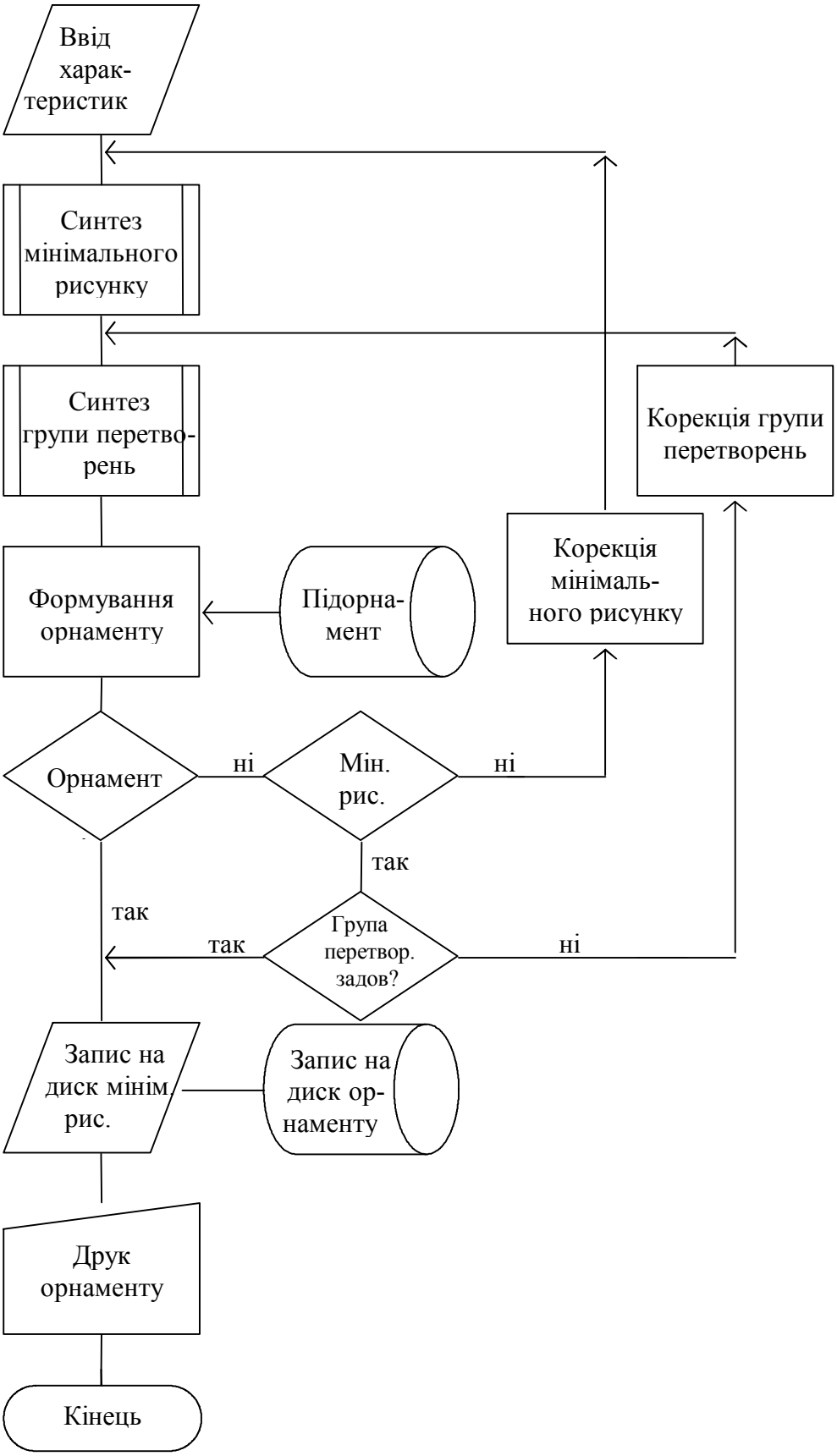


Рис. 3.50. Узагальнений алгоритм синтезу зображень-орнаментів.

4. На основі синтезованих підорнаментів формуємо орнамент. При цьому використовуємо підорнаменти з бази даних.

5. У випадку, коли сформований орнамент нас задовільняє, то проводимо запис його формули та формул використаних в ньому мінімальних рисунків на магнітний диск. При потребі виводимо орнамент на друк.

6. Коли синтезований орнамент нас не задовільняє, то проводимо його аналіз (розглядаємо мінімальні рисунки та групи перетворень над ними). Якщо використані мінімальні рисунки нас не задовільняють, проводимо їх корекцію (повторний синтез). Аналогічно, якщо групи перетворень не відповідають вимогам, проводимо їх корекцію (повторний синтез).

В приведений алгоритм входять алгоритми синтезу мінімального рисунку та синтезу груп перетворень. Розглянемо послідовно дані алгоритми.

Алгоритм синтезу мінімального рисунку наступний (рис. 3.51).

1. При синтезі мінімального рисунку можливі наступні варіанти:

-стохастичний синтез;

-детермінований синтез.

Крім цього можливо використовувати базу даних типових мінімальних рисунків (підрозділ 3.2). Першим кроком алгоритму є зчитування варіанту синтезу мінімального рисунку.

2. При використанні типових рисунків з бази даних переходимо на закінчення алгоритму.

3. У випадку використання стохастичного синтезу послідовно вибираємо по заданих законах розподілу дискретних випадкових величин (підрозділ 3.2) непохідні елементи, операції між ними та кольори непохідних елементів. Паралельно з випадковим формуванням рисунку проходить автоматичне формування формули мінімального рисунку.

4. При детермінованому синтезі задається аналітична формула згідно якої проходить формування мінімального рисунку (підрозділ 3.2).

5. Після закінчення синтезу формула рисунку записується в форматі *.txt, а сам мінімальний рисунок в форматі *.bmp.

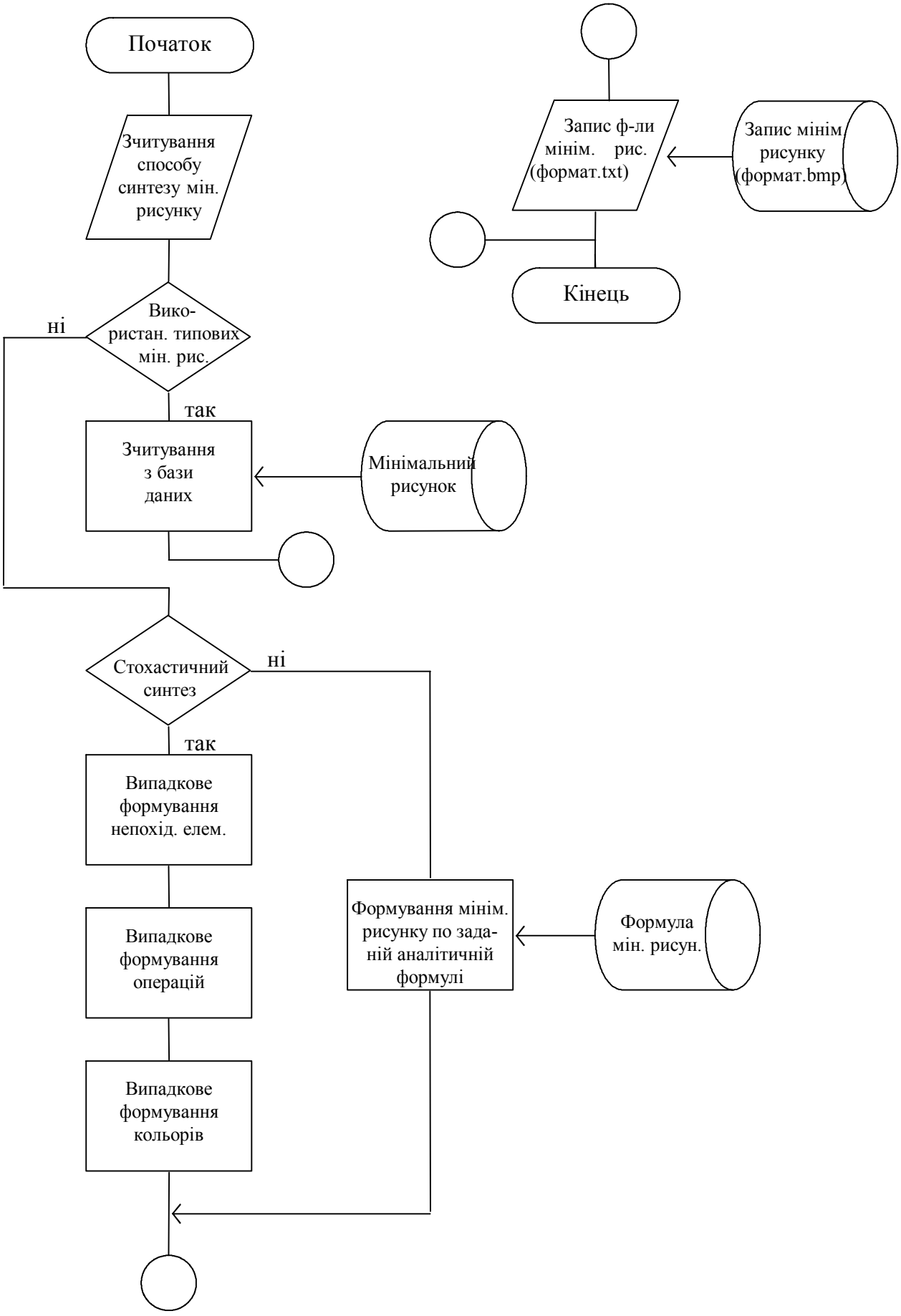


Рис. 3.51. Алгоритм синтезу мінімального рисунку.

Алгоритм синтезу груп перетворень (підорнаментів) наступний (рис.3.52):

1. Ввід групи перетворень та способу синтезу.
2. У випадку використання породжуючих перетворень синтез групи проводимо через них (підрозділ 3.1). При цьому використовуємо раніше сформований мінімальний рисунок .
3. При використанні рекурсивного підходу синтез групи здійснюємо рекурсивним способом (підрозділ 2.3).
4. Використовуючи осьові симетрії, на їх основі проводимо синтез заданої групи (підрозділ 3.1).
5. Формулу синтезованого підорнаменту (групи перетворень) записуємо в форматі txt на магнітний диск, а самого підорнаменту в форматі bmp.

Приведемо інформаційну модель побудови орнаменту (рис. 3.53).

Інформаційна модель побудови орнаменту передбачає використання (блок P_1) довідника “Види мінімальних рисунків” (необхідних для вибору варіантів мінімальних рисунків з метою формування підорнаментів орнаментистом). Крім цього, орнаментист має змогу (блок P_2) вибирати варіанти груп перетворень з довідника «Види груп перетворень» з метою створення нових підорнаментів. Наступний крок передбачає автоматичне формування підорнаменту на основі вибраного мінімального рисунку та групи перетворень (блок A_1). Орнаментист, використовуючи синтезований підорнамент та довіник «Види підорнаментів» формує остаточний варіант орнаменту (А2).

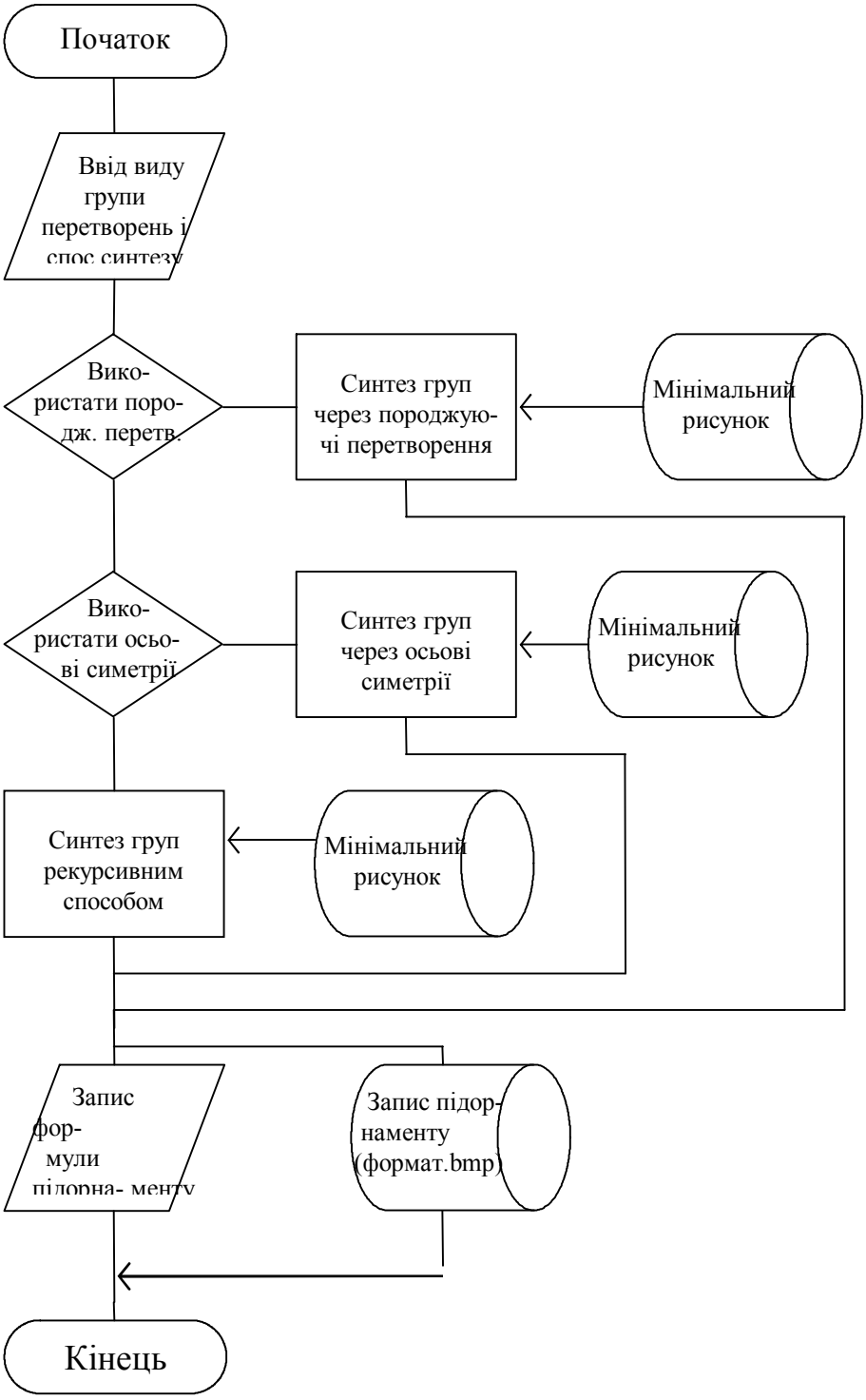


Рис. 3.52. Алгоритм синтезу груп перетворень.

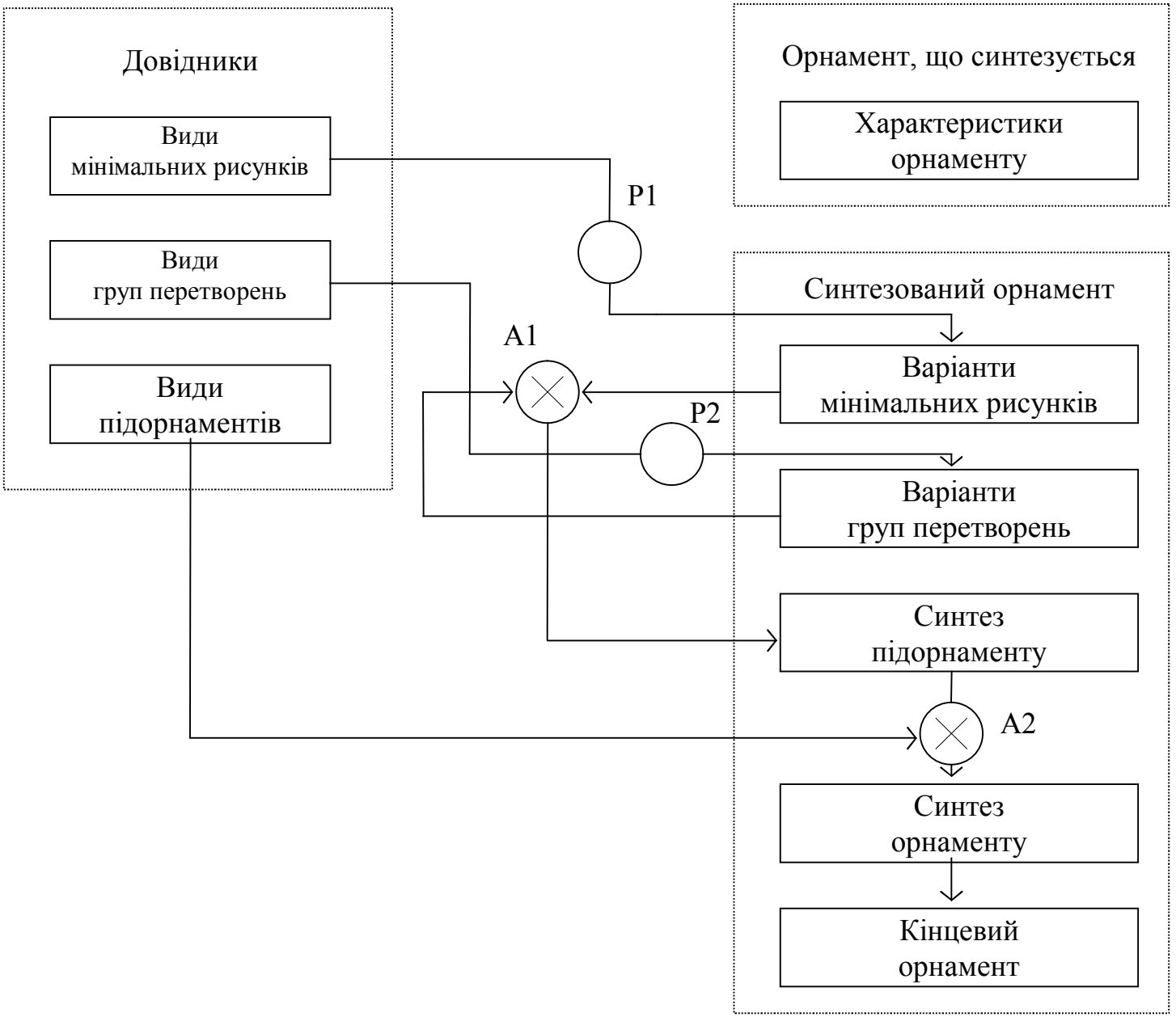


Рис. 3.53. Інформаційна модель побудови зображення-орнаменту

3.3.2. Оцінка ефективності архівування складних зображень-орнаментів.

В підрозділі 2.1, 1.2 показано, що симетричне зображення може бути представлене у вигляді орнаментів, підорнаментів, рапортів та мінімальних рисунків. Мінімальний рисунок входить до складу рапорту, рапорт до складу підорнаменту (симетричної групи перетворень), а орнамент складається з підорнаментів. Для складових зображення-орнаменту справедливі наступні твердження:

- рапорт для кожної групи різний;
- рапорт складається з одного або декількох мінімальних рисунків;
- підорнамент - це рапорт «плюс» перетворення над ним, який має обмежені геометричні границі;
- орнамент - сукупність підорнаментів.

Кожна із перерахованих складових може бути представлена з допомогою наступних форматів ∗.bmp та ∗.txt (з використанням формули побудови).

Оцінимо коефіцієнт стиснення для кожної із складових. Позначимо через $Q_{\min b}$ - об'єм пам'яті для збереження мінімального рисунку в ∗.bmp форматі. $Q_{\min t}$ - об'єм пам'яті для збереження мінімального рисунку в ∗.txt форматі. Тоді

$$\frac{Q_{\min b}}{Q_{\min t}} = K_{\min},$$

де $K_{\min}$ - коефіцієнт стиснення мінімального рисунку.

У випадку збереження мінімального рисунку в ∗.bmp форматі $K_{\min}$ = 1. Як було показано в підрозділі 2.3, рапорт складається з одного або декількох мінімальних рисунків, тобто в форматі ∗.bmp об'єм пам'яті для нього рівний

$$RP_{bi} = K_i \cdot Q_{\min b},$$

де K_i - кількість мінімальних рисунків в рапорті. Відповідно в форматі ∗.txt

$$RP_{ti} = K_i \cdot Q_{\min t}.$$

Слід відмітити, що в форматі ∗.txt об'єм пам'яті практично не зростає від

кількості мінімальних рисунків (у формулі опису для кодування кількості мінімальних рисунків відводиться один символ, який рівний одному байтові), тобто $RP_t \approx Q_{\min t}$. Отже, коефіцієнт стиснення для рапорту буде рівний

$$K_{RP} = \frac{K_i \cdot Q_{\min b}}{Q_{\min t}} = K_i \cdot K_{\min}.$$

Якщо в рапорті мінімальний рисунок зберігається в ∗.bmp форматі, то тоді коефіцієнт стиснення буде рівний

$$K_{RP} = \frac{K_i \cdot Q_{\min b}}{Q_{\min b}} = K_i.$$

Відповідно для підорнаменту об'єм пам'яті в форматі ∗.bmp буде рівний

$$POR_{bi} = RP_{bi} \cdot K_{IT} = K_i \cdot K_{IT} \cdot Q_{\min b}, \tag{3.16}$$

де K_{IT} - кількість ітерацій (кількість рапортів) в форматі ∗.txt.

$$POR_{ti} = RP_{ti} \cdot K_{IT} = K_i \cdot K_{IT} \cdot Q_{\min t}, \tag{3.17}$$

$$POR_{ti} \approx Q_{\min t}$$

(об'єм пам'яті для підорнаменту в форматі ∗.txt наближено рівний об'єму пам'яті мінімального рисунку). Тоді коефіцієнт стиснення для підорнаменту рівний

$$K_{PORi} = \frac{POR_{bi}}{POR_{ti}} \approx \frac{K_i \cdot K_{IT} Q_{\min b}}{Q_{\min t}} = K_i \cdot K_{IT} \cdot K_{\min}.$$

У випадку збереження мінімального рисунку в ∗.bmp форматі рапорту коефіцієнт стиснення буде рівний

$$K_{PORi} \approx \frac{K_i \cdot K_{IT} Q_{\min b}}{Q_{\min b}} = K_i \cdot K_{IT}.$$

У разі збереження рапорту в ∗.bmp форматі коефіцієнт стиснення рівний

$$K_{PORi} \approx \frac{K_i \cdot K_{IT} Q_{\min b}}{K_i \cdot Q_{\min b}} = K_{IT}.$$

Орнамент, як показано в підрозділі 1.2 складається з множини підорнаментів, тобто об'єм пам'яті для нього рівний

$$OR = POR_1 + POR_2 + \ldots + POR_n = \sum_{i=1}^{n} POR_i. \tag{3.18}$$

Підставивши у формулу 3.18 вирази POR з формул 3.16, 3.17 отримаємо

$$OR = RP_1 \cdot K_{IT1} + RP_2 \cdot K_{IT2} + ... + RP_n \cdot K_{ITn} = \sum_{i=1}^{n} RP_i \cdot K_{ITi} .$$

Тоді об'єм пам'яті для орнаменту в форматі ∗.bmp буде рівний

$$OR_b = RP_{b1} \cdot K_{IT1} + RP_{b2} \cdot K_{IT2} + ... + RP_{bn} \cdot K_{ITn} = \sum_{i=1}^{n} RP_{bi} \cdot K_{ITi}$$

в форматі ∗.txt (об'єм пам'яті рапортів наближено рівний об'єму пам'яті їх мінімальних рисунків, тобто не залежить від кількості ітерацій в рапорті)

$$OR_t = RP_{t1} + RP_{t2} + ... + RP_{tn} = \sum_{i=1}^{n} RP_{ti} \qquad (3.19)$$

Отже, коефіцієнт стиснення для орнаменту буде рівний

$$K_{OR} = \frac{\sum_{i=1}^{n} RP_{bi} \cdot K_{ITi}}{\sum_{i=1}^{n} RP_{ti}} \qquad (3.20)$$

Поскільки в форматі ∗.txt об'єм пам'яті для формули орнаменту наближено дорівнює сумі об'ємів пам'яті мінімальних рисунків підорнаментів, тобто $\sum_{i=1}^{n} RP_{ti} \approx \sum_{i=1}^{n} Q_{\min ti}$, тоді вираз для коефіцієнту стиснення орнаменту можна представити у вигляді

$$K_{OR} = \frac{\sum_{i=1}^{n} RP_{bi} \cdot K_{ITi}}{\sum_{i=1}^{n} Q_{\min ti}} .$$

Представимо графічно об'єми пам'яті для складових орнаменту (рис. 3.54). По осі абцис відкладемо об'єми пам'яті складових орнаменту в форматі ∗.txt, а по осі ординат об'єми пам'яті в форматі ∗.bmp. Із графіку видно, що відношення об'ємів пам'яті відповідно в форматах ∗.bmp та ∗.txt дасть коефіцієнт стиснення, причому коефіцієнт стиснення зростає пропорційно складовим орнаменту: мінімальний рисунок ($K_{\min}$), рапорт ($K_i \cdot K_{\min}$) та підорнамент ($K_i \cdot K_{IT} \cdot K_{\min}$) (криві 1, 2, 3).

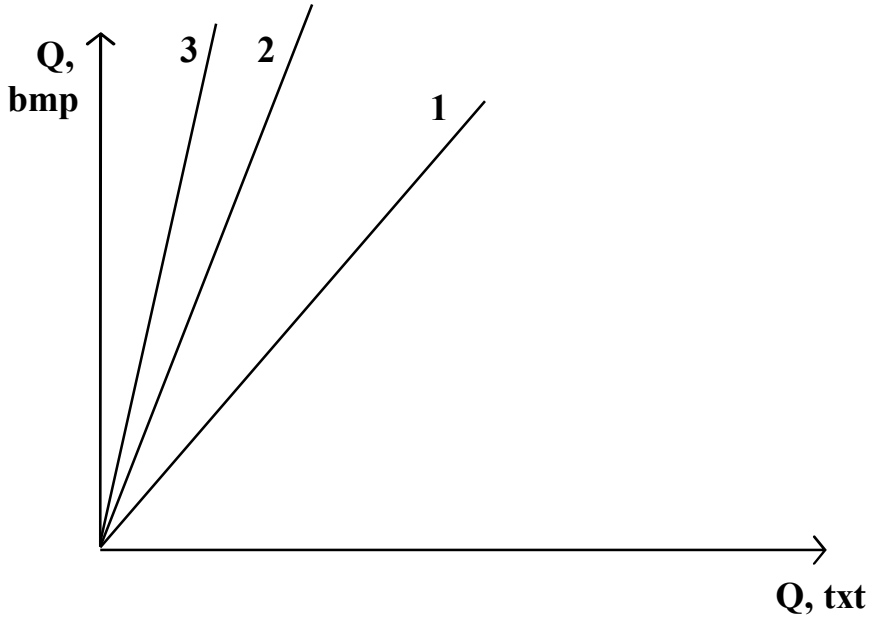


Рис. 3.54. Залежність об’ємів пам’яті в форматі bmp від txt для складових орнаменту.

Отже, в даному підрозділі запропонований узагальнений алгоритм синтезу зображень-орнаментів, їх моделювання, який базуються на алгоритмах формування мінімальних рисунків (детермінованим та стохастичним способами) та алгоритмах формування груп перетворень (детермінованим способом), розроблена інформаційна модель побудови орнаментів та проведено оцінку ефективності архівування симетричних зображень.

3.4.Основні результати, отримані у розділі 3

1.На основі використання породжувальних перетворень формалізовано групи перетворень на площині та смузі, що дало можливість синтезувати узагальнений алгоритм синтезу зображень-орнаментів.

2. Використовуючи запропоновану мову опису, розроблені алгоритми синтезу мінімального рисунку: розглянуто моделювання мінімального рисунку в різних функціональних базисах, доведена теорема про мінімальний базис, показані приклади їх застосування до опису мінімальних зображень та спроектована база даних.

3. Запропонований узагальнений алгоритм синтезу зображень-орнаментів, який базується на алгоритмах формування мінімальних рисунків (детермінованим та стохастичним способами) та алгоритмах формування груп перетворень (детермінованим способом), розроблена інформаційна модель побудови орнаментів, що дало можливість розробити редактор орнаментів та проведено оцінку ефективності архівування симетричних зображень.

РОЗДІЛ 4
РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ РЕДАКТОРІВ ЗОБРАЖЕНЬ-ОРНАМЕНТІВ І ВИШИВОК

4.1.Розробка редактора зображень-орнаментів

На основі запропонованої мови опису мінімального рисунку (підрозділ 2.1) та алгоритмів побудови орнаментальних груп (підрозділ 2.3, 3.1), розроблений редактор (генератор) зображень-орнаментів [29,30,33]. Даний програмний комплекс дозволяє синтезувати складні орнаментальні зображення на основі аналітичних формул мінімального рисунку, підорнаменту та орнаменту. Разом з тим, дана система дозволяє проводити архівування зображень шляхом виділення мінімального рисунку (формули в форматі txt або зображення в форматі bmp) та груп перетворень над ним, що здійснюється автоматизованим шляхом з участю людини-орнаментиста.

На основі розроблених інформаційної та даталогічної моделей (підрозділи 3.2, 3.3) синтезована програмна система.

Система складається з п’яти підсистем, кожна з яких реалізує окрему задачу, зокрема (рис. 4.1):

- детерміноване формування елементарного рисунку;
- стохастичне формування елементарного рисунку;
- формування орнаментальних груп;
- формування підорнаментів;
- вивід орнаменту на друк.

Розглянемо кожну із підсистем окремо з розглядом виконуваних ними функцій.

Підсистема детермінованого формування елементарного рисунку забезпечує наступні функції:

- набір та редагування формули зображення;
- трансляція формули зображення;
- графічне представлення формули зображення;

- зчитування та запис в БД формули мінімального рисунку та його зображення.

Підсистема стохастичного формування елементарного рисунку забезпечує наступні функції :

- автоматичне формування формули зображення по випадкових законах формування непохідних елементів, операцій над ними та їх кольорів;
- графічне представлення формули зображення;
- зчитування та запис в БД формули мінімального рисунку та його зображення.

Підсистема формування орнаментальних груп забезпечує наступні функції:

- автоматичне формування орнаментальної групи (підорнаменту) на основі заданої групи, використовуючи синтезований мінімальний рисунок;
- зчитування та запис в БД формули підорнаменту та його зображення.

Підсистема формування орнаменту забезпечує наступні функції:

- автоматизоване формування орнаменту на основі створених орнаментів;
- зчитування та запис в БД формули орнаменту та його зображення.

Підсистема виводу орнаменту на друк забезпечує наступні функції:

- вивід на друк цілого орнаменту;
- вивід на друк частини орнаменту;
- масштабування орнаменту.

Програмна система реалізована в середовищі мови програмування Borland Pascal with Objects 7.0 [103-105] під Windows 95.

Ієрархія процедур та модулів, які реалізують вище приведену структуру програмної системи зображена на рис. 4.1. Пояснимо приведені процедури (лістінги програм приведені в додатку Ж):

Процедура Raport служить для створення орнаментної групи (підорнаменту). Дана процедура, використовуючи створений мінімальний рисунок (на основі мови та алгоритмів створення мінімального рисунку (підрозділи 2.2, 3.2)) реалізує сімнадцять орнаментальних груп на площині та сім груп на смузі (на основі запропонованих алгоритмів (підрозділ 3.1).

Процедура Elem використовується для вводу нового підорнаменту, тобто для створення орнаменту із підорнаментів. Синтез орнаменту відбувається шляхом

послідовного заповнення площини синтезованими підорнаментами.

Процедура Srotate використовується для операції повороту мінімального рисунку і є допоміжною для процедури Raport.

Процедура MoveChoise служить для переміщення групи перетворень (підорнаменту) і є допоміжною для процедур Raport та Elem.

Процедура TMainFort.SaveBtnClick служить для запису орнаменту в БД.

Процедура TMainFort.LoadBtnClick використовується для завантаження орнаменту із БД.

Модуль Ornam3Z використовується для формування мінімального рисунку згідно мови опису та алгоритмів синтезу, запропонованих в підрозділах 2.2, 3.2. При цьому синтезований мінімальний рисунок може бути записаний в БД у форматі .txt (аналітична формула) та і в форматі .bmp (у вигляді зображення).

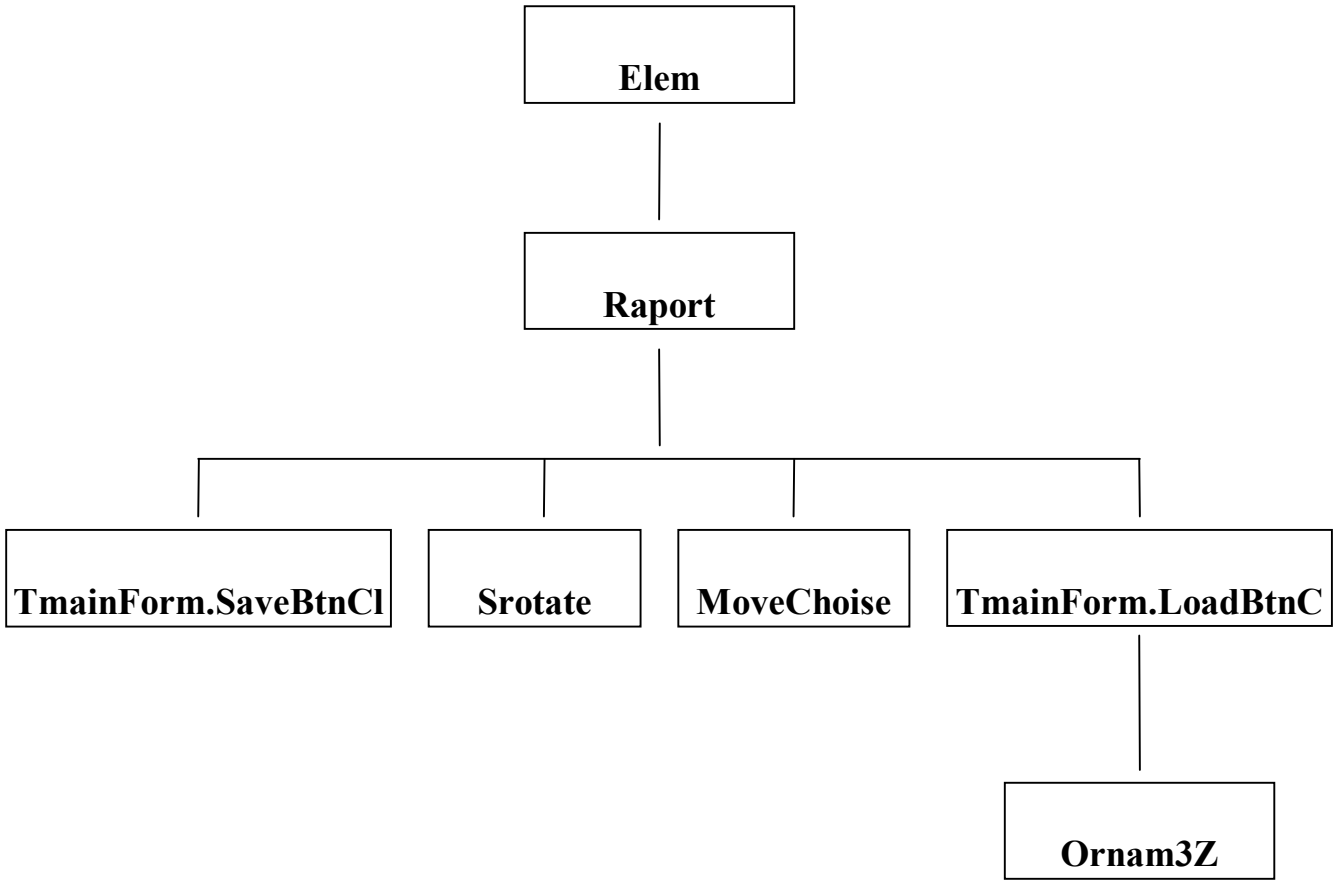


Рис. 4.1. Ієрархія процедур та модулів системи.

Опишемо алгоритм створення орнаменту за допомогою розробленого редактора (детальний опис інструкції користувача розробленого редактора приведений в додатку З). Щоб створити фрагмент орнаменту необхідно вибрати тип і розмір сітки. Для цього натискаємо кнопку «вибір сітки» (на правій панелі третя зверху (див. додаток З, рис.З.1) і в списку, що з'явився вибираємо потрібний тип сітки.

Мінімальний рисунок формується двома способами:

Синтезується (детерміновано, стохастично) новий рисунок;

Завантажується створений із бази даних.

Опишемо ці способи.

1) Якщо мінімальний рисунок не створений, то переходимо з головного вікна у вікно «малюнок» (рис. З.6 додатку З). Для цього натискаємо кнопку «малюнок» (на правій панелі - друга знизу). У наступному вікні встановлюємо розміри рисунка і натискаємо «Добре». Утворюється рисунок тільки відрізками заданого кольору по горизонталі, по вертикалі або по діагоналі “стьобанням" (див. множину непохідних елементів Е в підрозділі 2.2). Поточний колір встановлюється на лівій панелі. Можна залити поточним кольором область, обмежену точками інших кольорів (замкненим контуром).

Якщо при побудові рисунку допущені помилки або виникла необхідність змінити вид і разом з тим формулу рисунку, то для цього натискаємо кнопку «редагування формули». У вікні, що з'явилося набираємо або відредаговуємо формулу і натискаємо кнопку «добре».

2) Щоб вибрати вже створений потрібний мінімальний рисунок необхідно натиснути кнопку «вибір елемента» (на правій панелі четверта зверху), у вікні, що з'явилося вибрати один з двох варіантів:

- завантаження малюнку під його власний розмір;
- масштабування малюнку під розмір осередку сітки.

У наступному діалоговому вікні вибираємо файл малюнка лінійним розміром не більше 100×100 пікселів.

Після синтезу мінімального рисунку вибирається потрібна група перетворень

над його зображенням (тип заповнення). Для цього необхідно натиснути кнопку «вибір заповнення» (на правій панелі п'ята зверху), в вікні, що з'явилося вибрати з списку тип заповнення. Якщо вибраний тип відноситься до груп на смузі вибрати в наступному вікні розташування смуги: горизонтальне або вертикальне.

Вибравши параметри, натискаємо кнопку «робота з елементом» (рис. 3.2) і вказуємо мишею (натискаємо ліву кнопку) в тому осередку сітки, куди хочете вмістити початковий елемент групи. Якщо поточний тип сітки прямокутний, то у вікні, що з'явилося вибираємо кількість ітерацій (скільки разів початкова група буде розмножуватися у всі сторони). У випадку, коли всі параметри введені правильно натискаємо «Добре», якщо необхідно змінити параметри - «Відміна».

При потребі змінити розташування групи вибираємо кнопку «перемістити» (на верхній панелі восьма зліва), потім натискаємо правою кнопкою миші на потрібній групі, поки не буде вибрана саме вона (група вибрана тоді, коли її елементарні малюнки стали чорно-білими).

Якщо потрібно переробити останню групу натискаємо кнопку «знищити останнє» (рис. 3.2) і будуємо групу наново.

У орнамент можна вставляти текстові рядки. Для цього натискаємо кнопку «мітка» (на верхній панелі шоста праворуч), натисненням миші вставляємо смугу рядка і вводимо текст. По закінченні введення рядка натискаємо “Enter". У випадку необхідності зміни шрифта двічі натискаємо мишею на кнопці «мітка» і вибираємо в діалоговому вікні, що відкрилося, параметри шрифту.

Описану послідовність дій можна повторювати багато разів, поки орнамент не буде побудований.

Створений орнамент можна зберегти в файл розширення *.orn з допомогою кнопки «зберегти орнамент» (на правій панелі верхня, рис. 3.9).

Збережений орнамент завантажується по натисненню кнопки «відкрити орнамент» (на правій панелі друга зверху, рис. 3.9).

Орнамент можна очистити, натиснувши кнопку «очистити» (на верхній панелі п'ята праворуч, рис. 3.2).

Для побудованого орнаменту програма може побудувати його формулу. Щоб переглянути її натискаємо кнопку «формула» (на верхній панелі перша праворуч). Щоб зберегти формулу в текстовий файл (з розширенням *.txt) вибираємо мишею на пункті меню «зберегти формулу» в розділі «файл» або натискуємо “Shift+F2".

Щоб роздрукувати орнамент натискаємо кнопку «друк» (на верхній панелі - друга праворуч). Якщо потрібно роздрукувати весь орнамент вибираємо у вікні, що з'явилося, опцію «друк всього орнаменту» і натискаємо кнопку «почати друк». Якщо потрібно роздрукувати фрагмент вибираємо опцію «друк прямокутного фрагменту» і натискаємо кнопку «відмітити фрагмент». Після цього в основному вікні натискаємо кнопку миші, розтягуємо мишею відмічений штриховкою прямокутник фрагмента і відпускаємо кнопку миші. Якщо відмічений фрагмент задовільний то натискаємо кнопку «почати друк», якщо не задовільний то знов натискаємо «відмітити фрагмент». У випадку необхідності відмінити роздруковування, натискаємо «відміна друку». Для попереднього перегляду орнаменту натискаємо кнопку «перегляд». Для повернення до основного вікна потрібно натиснути будь-яку клавішу або натиснути кнопкою миші.

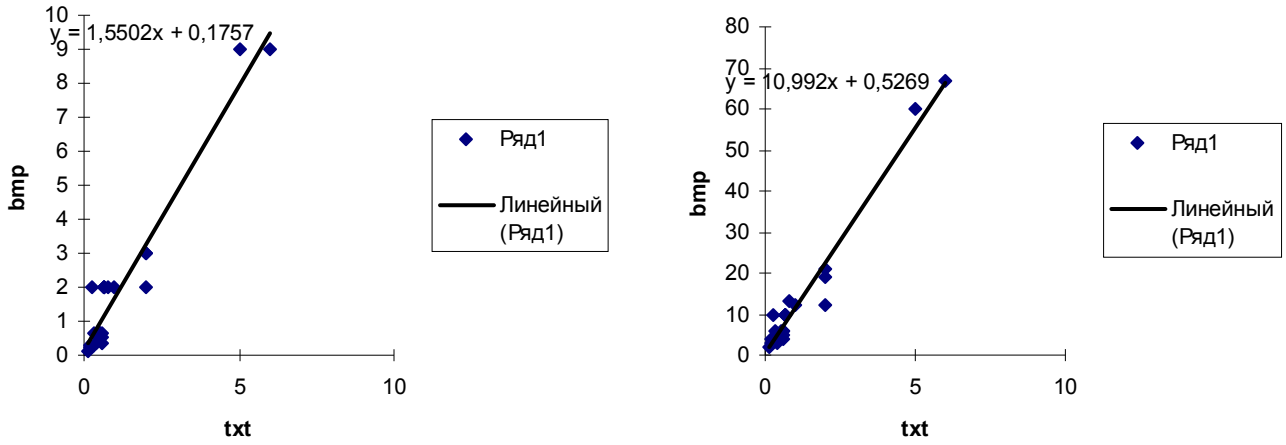
Приклади синтезованих орнаментів приведені в додатку Е. Проведемо аналіз (оцінку) архівування зображень за допомогою розробленого редактора. У таблиці 4.1 показані об’єми пам’яті для мінімальних рисунків та підорнаментів відповідно у форматах ∗.bmp та ∗.txt при заданому числі ітерацій 3 та групі перетворень - pg. Здійснений аналіз у середовищі Microsoft Excel 7.0 показує, що середній коефіцієнт стиснення мінімальних рисунків становить 5.8, а середній коефіцієнт стиснення для підорнаментів рівний 24.8. Графіки проведеного аналізу показані на рис. 4.2.а та рис. 4.2.б.

Отже, в даному підрозділі описано розроблений редактор орнаментів, який програмно релізований на основі запропонованих алгоритмів синтезу груп перетворень та мінімальних рисунків, що дало можливість економно зберігати наявні орнаменти та створювати нові та зроблено оцінку архівування складних зображень-орнаментів.

Таблиця 4.1

Об’єми пам’яті (Кб) складових орнаменту (група pg) в форматах *.bmp, *.txt

Код мінімальн. рисунку	Тип складових орнаменту					
	Мінімальний рисунок		Рапорт		Підорнамент	
	*.bmp	*.txt	*.bmp	*.txt	*.bmp	*.txt
MRB1A	3	2	6,8	2	21	2
MRB1Б	2	0,256	3,2	0,287	10	0,287
MRB1B	0,374	0,579	1	0,601	4	0,601
MRB1Г	9	6	30	6	67	6
MRB1Д	0,67	0,536	1,8	0,576	6	0,576
MRB1E	0,134	0,107	0,6	0,138	2	0,138
MRB1Є	2	0,664	3,2	0,695	10	0,695
MRB1Ж	0,558	0,569	1,6	0,598	5	0,598
MRB2A	0,67	0,549	2	0,58	6	0,58
MRB2Б	2	0,951	4	0,982	12	0,982
MRB2B	2	0,758	4,2	0,789	13	0,789
MRB2Г	3	2	6	2	19	2
MRB2Д	0,222	0,201	1	0,232	3	0,232
MRB2E	2	0,621	3,2	0,652	10	0,652
MRB2Є	0,306	0,167	1	0,198	4	0,198
MRB2Ж	0,67	0,334	2	0,365	6	0,365
MRB2З	0,366	0,377	1,6	0,408	5	0,408
MRB2И	0,27	0,342	1	0,371	3	0,371
MRB3A	2	0,652	3,2	0,683	10	0,683
MRB3Б	0,67	0,524	2	0,537	6	0,537
MRB3B	2	2	4	2	12	2
MRB3Г	9	5	18	5	60	5



а) б)

Рис. 4.2. Залежність об’ємів пам’яті в форматі bmp від txt для мінімальних рисунків (додаток В) (а) та підорнаментів (на основі групи pg) (б).

4.2. Розробка програмно-апаратного комплексу створення вишивок

Описаний в попередньому підрозділі комплекс програм «Редактор орнаментів» дозволяє створювати складні зображення- орнаменти, наприклад, українські народні вишивки (підрозділ 1.2). Сучасні швейно-вишивальні машини (наприклад, фірми «BROTHER») дозволяють програмувати зображення на картріджи з метою автоматичної вишивки. Але для цього необхідно проредагувати синтезовану вишивку та записати її на носій в форматах фірми. Для реалізації цих функцій був спроектований програмно-апаратний комплекс створення та редагування вишивок [33]. Він дозволяє створювати самостійно різні вишивки для швейних машин фірми моделей «PE -100» і «Super Galaxie». В склад комплексу входить пакет прикладних програм та пристрій вводу-виводу інформації (програматор) для репрограмуючого картріджу «BROTHER».

При проектуванні редактора вишивок було використоно об’єктно-орієнтований підхід [106, 107]. Розвиток та застосування цього підходу дозволяє на якісно новому рівні розв’язувати проблеми створення програмного забезпечення в

стислі терміни невеликими групами розробників. В результаті аналізу системи редагування вишивок на першому етапі проектування було виділено наступні об’єкти із своїми властивостями:

1. Файл.
2. Картинка.
3. Область.
4. Стібки.

Інформаційна модель редактора вишивок представлена на рис. 4.3. Другий етап об’єктно-орієнтованого проектування передбачає розробку діаграм переходів для об’єктів, що складають інформаційну модель. На рис.4.4 представлена діаграма переходів для об’єкту «Картинки». Аналогічно будуються діаграми переходів для інших об’єктів, використовуючи при цьому їхні властивості.

Редактор вишивок реалізований в середовищі мови програмування Borland Pascal with Objects 7.0 [103] під Windows 95. Як відомо, з точки зору програміста, Windows представляє набір підпрограм, користуючись якими можна ефективно створювати інтерфейс зі всіма характерними для Windows атрибутами (вікнами, кнопками, смугами скролінгу). Прикладні програми під час своєї роботи використовують функції та процедури операційного середовища (інтерфейс прикладної програми - API). Будь-які Windows - додатки базуються на об’єктній технології. Для спрощення розробки програм в даній мові програмування застосовується бібліотека OWL (Object Library) - об’єктно-орієнтована надбудова над Windows API. Крім цього будь-яка Windows програма виводить інформацію на графічні пристрої, використовуючи спеціальні функції, які утворюють GDI - інтерфейс.

Для запису спроектованої вишивки на картрідж фірми «BROTHER» розроблений апаратний комплекс (рис.4.5), який побудований по двохрівневій структурі і складається з ПЕОМ (типу IBM PC) як другого рівня і програматора як першого рівня. При проектуванні даного апаратного комплексу використано методику та елементну базу відображену в джерелах [108-112].

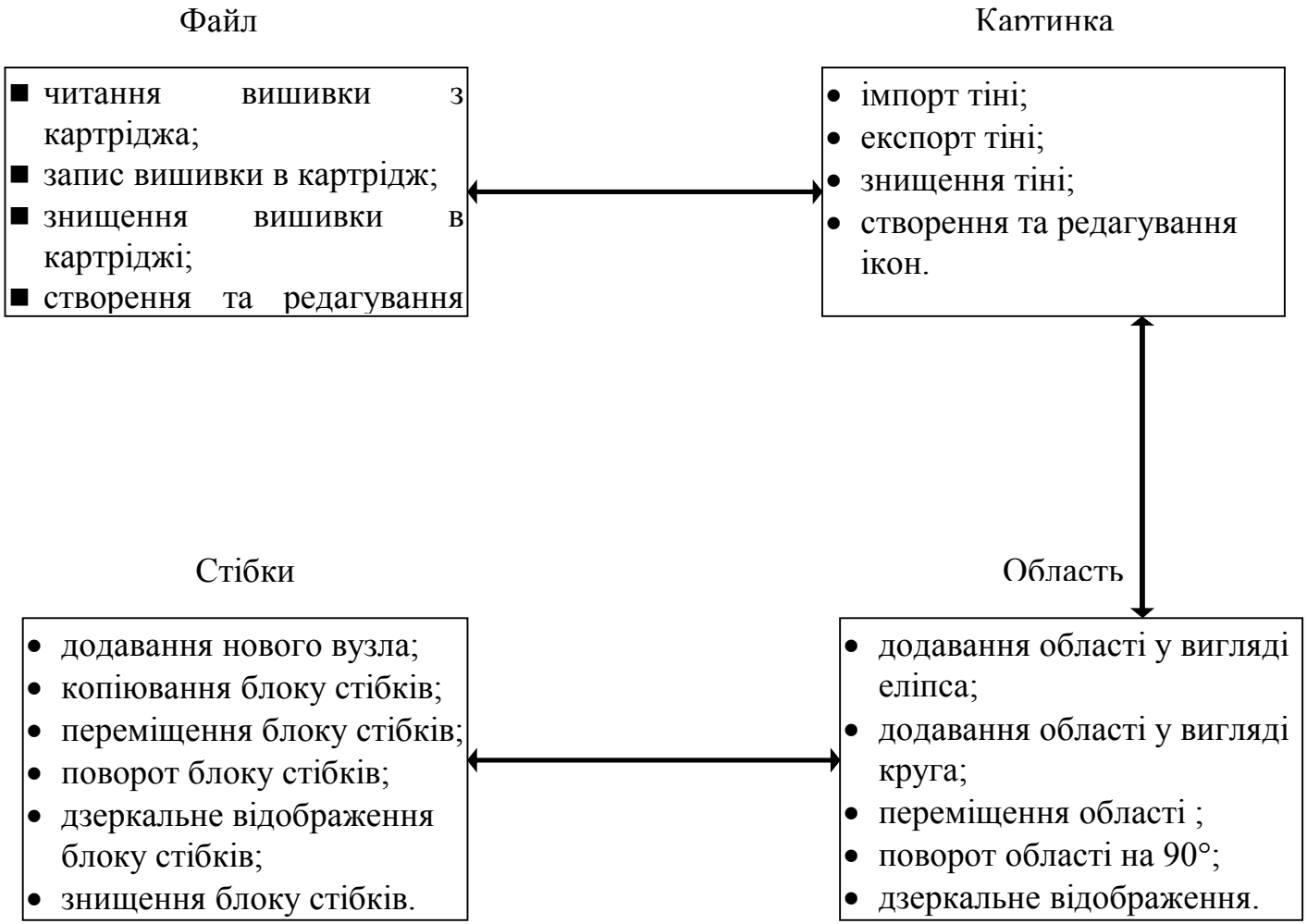


Рис. 4.3. Інформаційна модель «Редактора вишивок».

Рис. 4.4. Діаграма переходу в стани об’єкту «Картинка».

ПЕОМ і програматор з'єднані між собою послідовним інтерфейсом типу RS-232C [97]. ПЕОМ має наступну конфігурацію: центральний процесор, дисплей, клавіатуара, накопичувач на гнучких магнітних дисках НГМД з контролером. Програматор побудований на базі однокристальної мікро-ЕОМ (ОМЕОМ) типу КМ1816ВЕ31 (рис.4.5), на вхід якої через блок гальванічної розв’язки підключений інтерфейс RS-232C. В зв’язку з необхідністю кодувати отримане від редактора вишивок зображення в форматах «BROTHER» було збільшено об'єм постійного запам'ятовуючого пристрою (ПЗП) до десяти кБ (дві мікросхеми К573РФ6). З іншої сторони необхідно зберігати поточні розкладені на стібки зображення. Це зумовило збільшення об'єму оперативного запам'ятовуючого (ОЗП) пристрою (мікросхема К573РУ10). Програматор працює наступним чином. Зображення за допомогою «Редактора вишивок» перетворюється в масив стібків. Даний масив через

послідовний інтерфейс RS-232C передається в ОЗП програматора, де за допомогою програм перекодування, записаних в ОЗП, перетворюється в масив формату «BROTHER». По команді із ПЕОМ перекодований масив записується в катрідж. За його допомогою швейні машини моделей «PE -100» і «Super Galaxie» здійснюють в автоматичному режимі вишивання запрограмованого зображення.

Редактор вишивок, який записаний в ПЕОМ виконує наступні функції:

- створення картинки для проектованої вишивки в форматі *.bmp;
- створення необхідних контурів для автоматичного заповнення областей стібками;
- автоматичне заповнення областей і обшивка контурів стібками із встановленими параметрами;
- редагування результатів автоматичного заповнення;
- ручне накладання стібків;
- створення шарів вишивки.

Програматор виконує наступні функції:

- читання даних із картріджа;
- запис даних в картрідж;
- визначення серійного номеру картріджу;
- стирання даних із катріджу;
- шифрування даних;
- дешифрування даних;
- здійснення захисту даних.

Апаратні вимоги до програмно-апаратного комплексу створення вишивок наступні:

Комп’ютер..... IBM або сумісний

Процесор..... 80486

Операційна система..... Windows 3.1 і вище

Ємність оперативної пам’яті..... 8 Мбайт і вище

Ємність дискового простору.....4 Мбайт і вище

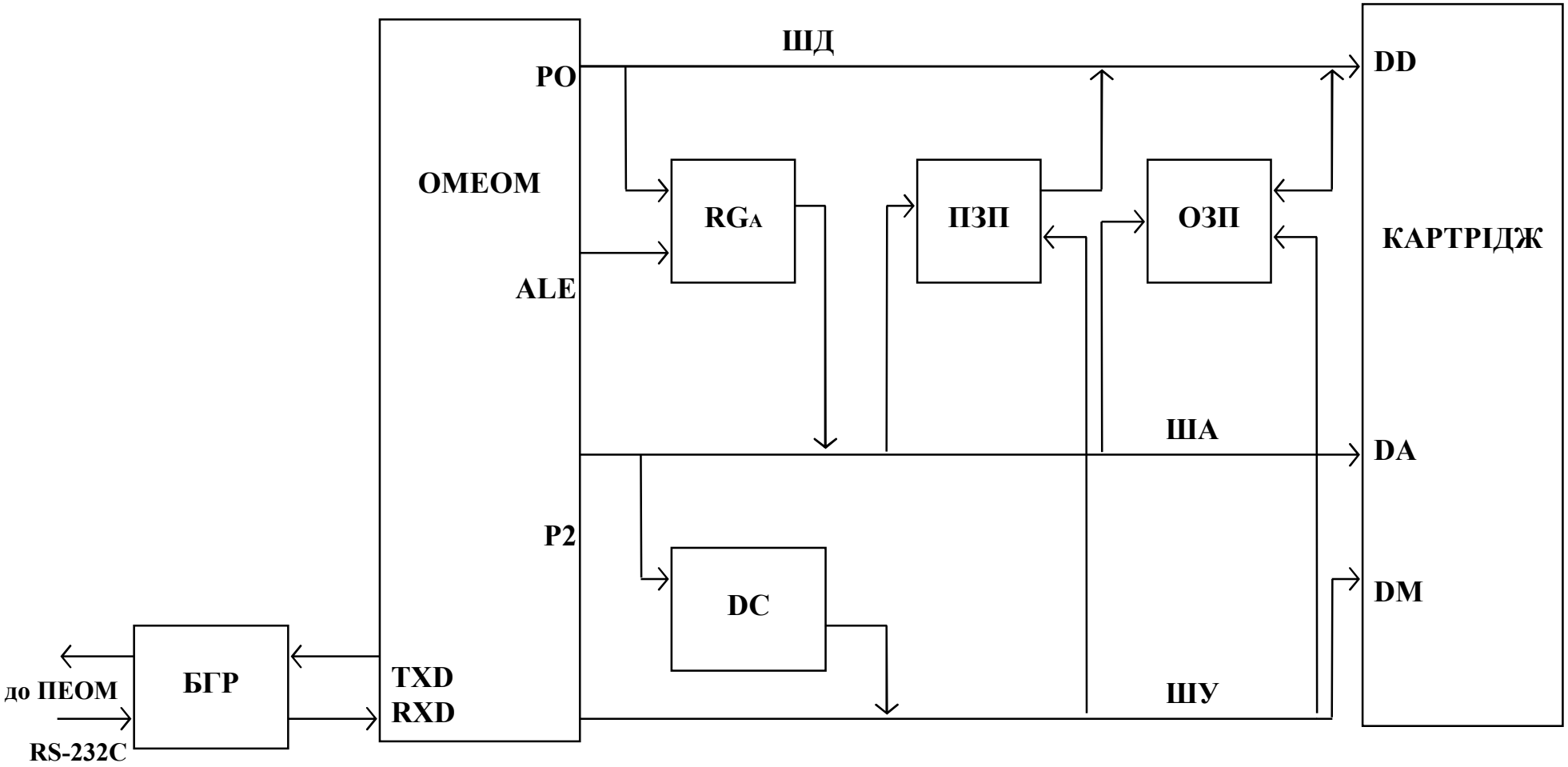


Рис. 4.5. Структурно-функціональна схема комплексу створення вишивок

Тип дисплею.....SVGA

Системний вхід.....RS-232C

Прінтер.....графічний

Таким чином, використання розробленого програмно-апаратного комплексу створення вишивок дозволило автоматизувати процес створення та редагування вишивок для відтворення їх в натуральному вигляді (на тканині).

4.3. Основні результати, отримані у розділі 4

Розроблений редактор зображень-орнаментів, який програмно релізований на основі запропонованих алгоритмів синтезу груп перетворень та мінімальних рисунків, дав можливість економно зберігати наявні зображення-орнаменти та створювати нові.

2. Використання розробленого програмно-апаратного комплексу створення вишивок дозволило автоматизувати процес створення та редагування вишивок для відтворення їх в натуральному вигляді (на тканині).

3. Спроектовані та впроваджені редактори зображень-орнаментів та вишивок стали ядром нової інформаційної технології побудови, зберігання та виготовлення складних зображень в легкій, текстильній промисловостях та поліграфії.

ВИСНОВКИ

1. Показано фундаментальну роль поняття симетрії (в науці, техніці, мистецтві, побуті), розкрито її геометричну суть та приведено групи перетворень симетричних зображень для площини та смуги.
2. На основі поняття симетрії розглянуто українську народну вишивку, як об’єкт, наділений структурою, виділено структурні компоненти орнаменту та його характеристики. Показано, що перетворення над зображеннями-орнаментами (вишивками) утворюють 12 груп перетворень на площині та 7 груп перетворень на смузі, які входять до відповідних федорівських груп.
3. Проаналізовано основні методи опису, синтезу, моделювання та архівування складних зображень і програмні засоби їх реалізації, обгрунтовано застосування структурних методів при описі зображень-орнаментів.
4. Розроблено метод опису та моделювання складних симетричних зображень-орнаментів, який базується на алгоритмах формалізації груп перетворень, мові опису мінімальних рисунків та алгоритмах формалізації рапортів.
5. На основі запропонованої мови опису мінімального рисунку і груп перетворень здійснено моделювання складних зображень-орнаментів.
6. Використовуючи породжувальні перетворення, формалізовано групи перетворень на площині та смузі, що дало можливість змоделювати узагальнений алгоритм синтезу зображень-орнаментів.
7. На основі запропонованої мови опису зображень, розроблені алгоритми синтезу мінімального рисунку: здійснено моделювання мінімального рисунку в різних

функціональних базисах, доведено теорему про мінімальний базис, показано приклади їх застосування до опису мінімальних зображень та спроектовано базу даних.

8. Запропоновано узагальнений алгоритм синтезу зображень-орнаментів, який базується на алгоритмах формування мінімальних рисунків (детермінованим та стохастичним способами) та алгоритмах формування груп перетворень (детермінованим способом); розроблено інформаційну модель побудови орнаментів, що дало можливість розробити редактор зображень-орнаментів та дано оцінку ефективності архівування симетричних зображень.
9. Розроблені редактори зображень-орнаментів та вишивок дозволили створювати нові та економно зберігати наявні зображення-орнаменти, автоматизувати процес створення та редагування вишивок для відтворення їх в натуральному вигляді (на тканині).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Урманцев Ю.А. Симметрия природы и природа симметрии. - М.: Мысль, 1974. - 229 с.
2. Вейль Г. Симметрия: Пер. с англ.-М.: Наука, 1968. - 191 с.
3. Вернадский В. И. Размышления натуралиста: В 2 кн. - М.: Наука, 1975. -Кн. 1: Пространство и время в неживой и живой природе. - 174 с.
4. Фейнман Р. Характер физических законов: Пер. с англ. - 2-е изд., испр. - М.: Наука, 1987. - 160 с.
5. Шубников А.В., Копцик В.А. Симметрия в науке и искусстве. - М.: Наука, 1972. - 339 с.
6. Шафрановский И.И. Симметрия в природе. - Л.: Недра, 1968. - 184 с.
7. Федоров Е.С.Симметрия и структура кристаллов: Основные работы.-М.: Изд-во Акад. наук СССР, 1949. - 631 с.
8. Ивахненко А.Г. Самообучающиеся системы распознавания и автоматического управления. - К.: Техніка, - 1969. - 512 с.
9. Грицык В.В. Распараллеливание алгоритмов обработки информации в системах реального времени. - К.: Наукова думка, 1981. - 214 с.
10. Сироджа И.Б., Тупало В.Г., Левин С.В. Структурно-аналитические модели и алгоритмы распознавания и идентификации объектов управления. - К.: Техніка, 1993. - 205 с.
11. Вапник В.Н., Червоненкис А.Я. Теория распознавания образов (статистические проблемы обучения). - М.: Наука, 1974. - 416 с.
12. Горелик А.Л., Скрипкин В.А. Методы распознавания. - М.: Высшая школа, 1989. - 232 с.
13. Анисимов Б.В., Курганов В.Д., Злобин В.К. Распознавание и цифровая обработка изображений: Учеб. пособие для студентов вузов. - М.: Высш. шк., 1983. - 295 с.
14. Александров В.В., Горский Н.Д. Представление и обработка изображений. Рекурсивный подход.-Л.:Наука, 1985.-190с.

15.Шикин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистические изображения. - М.: ДИАЛОГ-МИФИ, 1995. - 288 с.

16.Корриган Дж. Компьютерная графика: Секреты и решения : Пер. с англ. - М.: Энтроп, 1995. - 352 с.

17.Галузинський Г.П., Гордієнко І.В. Сучасні технологічні засоби обробки інформації: Навч. посібник. - К.:КНЕУ, 1998. - 224 с.

18.Иванов В.П., Батраков А.С. Трехмерная компьютерная графика / Под ред. Г.М.Полищука. - М.: Радио и связь, 1995. - 224 с.

19.Григорьев А. Графика на экране. - М.: Диалог - Мифи, 1992.- 79 с.

20.Мичи Д., Джонстон Р. Компьютер-творец: Пер. с англ. - М.: Мир, 1987. - 255с.

21.Хирн Д., Бейкер М. Микрокомпьютерная графика: Пер. с англ. - М.: Мир, 1987. - 352с.

22.Зенкин А.А. Когнитивная компьютерная графика / Под ред. Д.А.Поспелова. - М.:Наука, Гл. ред. физ.-мат. лит., 1991.- 192 с.

23.Искусственный интеллект: В 3-х кн. / Справочник / Под ред. Д.А. Поспелова. - М.: Радио и связь, 1990. - Кн. 2: Модели и методы. - 304 с.

24.Фу К. Структурные методы в распознавании образов: Пер. с англ. - М.: Мир, 1977. - 320 с.

25. Александров В.В., Горский Н.Д. Алгоритмы и программы структурного метода обработки данных. - Л.: Наука, 1983. - 208 с.

26.Миллер Дебора CorelDRAW 8. Библия пользователя: Пер. с англ. - К.; М.; СПб.: «Диалектика», 1998. - 624 с.

27. Полонская Е. Настоящая графика? - Это очень просто // Компьютеры + Программы. - 1999. - №5. - С. 14-21.

28.Березька К.М., Карпінський М.П. Мова опису складних зображень // Автоматика. Автоматизация. Электротехнические комплексы и системы. - 1997. - №1. - С.30-37.

29.Карпінський М.П., Березька К.М., Березький О.М. Генератор симетричних зображень // Автоматика. Автоматизация. Электротехнические комплексы и

системы. - 1998. - №1. - С.152-157.

30. Березька К.М., Березький О.М. Автоматизоване генерування орнаментних зображень // Вісник ДУ «Львівська Політехніка». Прикладна математика. - 1998. - № 337. - С.158 - 160.

31. Березька К.М., Березький О.М. Рекурсивні алгоритми генерування складних зображень // Матеріали Міжнародної наукової конференції «Сучасні проблеми математики». - Чернівці, 1998. - Част. 1. - С. 44-46.

32. Березька К.М., Березький О.М. Плоскі федорівські групи в складних симетричних зображеннях // Матеріали Всеукраїнської наукової конференції «Застосування обчислювальної техніки, математичного моделювання та математичних методів у наукових дослідженнях». - Львів, 1997. - С. 11.

33. Березька К.М., Березький О.М. Програмно-апаратний комплекс створення вишивок // Матеріали Всеукраїнської наукової конференції «Застосування обчислювальної техніки, математичного моделювання та математичних методів у наукових дослідженнях». - Львів, 1999. - С. 7.

34. Узоры симметрии: Пер с англ. / Под редакцией М.Сенешаль и Дж. Флека. - М.: Мир, 1980. - 272 с.

35. Кокстер Г.М. Введение в геометрию: Пер. с англ.-М.: Наука, 1966. - 648 с.

36. Бахман Ф. Построение геометрии на основе понятия симметрии: Пер. с нем. - М.: Наука, 1969. - 380 стр.

37. Шугаев В.М. Орнамент на ткани (теория и методика построения). - М.: Легкая индустрия, 1969. - 88 с.

38. Рудинская А.И. Орнамент: Искусство математика и художника. - Днепропетровск: Изд-во ДГУ, 1994. - 112с.

39. Рыбаков Б.А. Язычество древних славян.-М.: Наука, 1981. - 606 с.

40. Герчук Ю. Структура и смысл орнамента // Декоративное искусство СССР.- 1979. - №1. - С.30-33.

41. Михайленко В.Е., Кащенко А.В. Природа - геометрия - архитектура. - К.: Будівельник, 1981. - 184 с.

42. Найден О.С. Орнамент українського народного розпису. (АН УРСР. Інститут мистецтвознавства, фольклору та етнографії ім. М.Т. Рильського). - К.: Наук. думка, 1989. - 136 с.

43. Захарчук-Чугай Р.В.Українська народна вишивка.-Київ: Наукова думка, 1988. - 192 с.

44. Кара-Васильєва Т. Українська вишивка: Альбом. - К.: Мистецтво, 1993.- 264 с.

45. Знаки (155 стародавніх українських вишивок) / Упорядник Островська Т.О.- Київ: Молодь, Бібліотечка журналу "Соняшник", 1992. - 72 с.

46.Запаско Я.П. Орнаментальне оформлення української рукописної книги. - К.: Вид-во Акад. наук УРСР, 1960. - 183 с.

47.Коксетер Г.С.М., Мозер У.О.Дж. Порождающие элементы и определяющие соотношения дискретных групп: Пер. с англ.- М.: Наука, 1980. - 240 с.

48.Свирин А.И. Древнерусское шитье. - М.: Искусство, 1963. - 152 с.

49. Матейко К.М. Український народний одяг.- К.: Наук. думка, 1977. - 224 с.

50. Николаева Т.А. Украинская народная одежда. Среднее Поднепровье. - К.: Наукова думка, 1987. - 248 с.

51.Кислюк О.С. MORPHO - программная система моделирования развития геометрических форм растений // Программирование. - 1995. - №5. - С.55-62.

52. Кислюк О.С. Формальная система, порождающая пейзажи в стиле китайской монохромной средневековой живописи // Программирование. - 1996. - №2. - С.53-63.

53. Сундучков А. Фракталы и синтез изображений // Компьютеры + Программы. - 1996. - №6(30).- С.27-31.

54. Гузик В.Ф., Костюк А.И. Фрактальное сжатие и восстановление изображений // УСиМ. - 1998. - № 1. - С. 44-49.

55.Grabska E. Theoretical concepts of graphical modeling. Part one: realization of CP-graphs // Machine graphics & vision. - 1993.-Vol.2, № 1.- P. 3-38.

56. Grabska E., Grabski W., Jablonski M., Skomorowski M. Application of neural networks to pattern generation // Machine graphics & vision.-1994.-Vol.3, Nos. 1/2.- P.

161-170.

57. Головко В.А. Нейроинтеллект: теория и применение. Книга 1: Организация и обучение нейронных сетей с прямым и обратными связями. - Брест: Изд. БПИ, 1999. - 264 с.

58. Szyszkowicz M. Self-similar structures generated by the logistic map // Machine graphics & vision.-1993.-Vol.2, № 1.- P. 95-101.

59. Szyszkowicz M. Pictures of chaos // Machine graphics & vision.-1993.-Vol.2, № 3.- P. 263-270.

60. Березька К.М. Структурний метод опису складних симетричних зображень // Матеріали Всеукраїнської наукової конференції «Застосування обчислювальної техніки, математичного моделювання та математичних методів у наукових дослідженнях». - Львів, 1997. - С. 10-11.

61. D.Tveter. The Pattern Recognition Basis of Artificial Intelligence. IEEE Computer Society, 1997. - 350 p.

62. Закревский А.Д., Романов В.И. Реализация логического подхода к пословному распознаванию речи // УСиМ. - 1996. - № 6. - С. 16-19.

63. Хомский Н. Формальные свойства грамматик // "Кибернетический сборник".- М.: Мир. - 1966. - новая серия, Выпуск 2. - С. 121-230.

64. Мальцев А.И. Алгоритмы и рекурсивные функции. - М.: Наука, 1986. - 367 с.

65. Storer J.A. Data Compression: Methods and Theory. - USA: Computer Science Press, 1988.

66. Королев А.В., Рубан И.В. Сжатие видеоданных сериями граничных элементов // Электронное моделирование. - 1997.- № 5. - С. 31-40.

67. Skarbek W. Methods of digital image archivization. Part three: compressing images // Machine graphics & vision. - 1993.-Vol.2, № 1.- P. 53-86.

68. Королев А.В., Рубан И.В., Малахов С.В. Метод сжатия видеоданных посредством преобразований // Электронное моделирование. - 1999. - № 4. - С.47-56.

69. Зелов С. Цифровое кодирование видеоизображений // Компьютер Пресс. - 1997. - №3. - С.172-178.

70. Чак Хендерсон. Посредники изображений: маленькие гиганты компьютерной графики // Компьютеры + Программы. - 1997. - №1-2. - С. 24-29.

71. Петренко Д.А. Разработка приложений с помощью системы Authorware Professional // Компьютеры + Программы. - 1994. - №2. - С. 34-38.

72. Иванов В.В. Методы вычислений на ЭВМ: Справочное пособие. - К.: Наукова думка, 1986. - 584 с.

73. Шрейдер Ю.А. Равенство, сходство, порядок. - М.: Наука, 1971. - 256 с.

74. Гроссман И., Магнус В. Группы и их графы: Пер. с англ. - М.: Мир, 1971. - 246 с.

75. Березька К.М. Структурний метод опису складних зображень (на прикладах української вишивки) // Вісник ДУ «Львівська політехніка». Комп'ютерна інженерія та інформаційні технології. - 1998. - № 351. - С. 172 - 177.

76. Мальцев А.И. Группы и другие алгебраические системы // Математика, ее содержание, методы и значение.- М.: Изд-во АН СССР, 1956. - Т.3. - С. 269-273.

77. Стоян Ю.Г. Размещение геометрических объектов. - К.: Наукова думка, 1975. - 240 с.

78. Гусев А.А., Смирнова И.М. Языки, грамматики и абстрактные автоматные модели // Автоматика и телемеханика.- 1968.-№4.- С.72-94.

79. Shaw A.C. A formal picture description scheme as a basis for picture processing system // Information and Control. - 1969. - Vol.14, № 1. - P. 9-52.

80. Березька К.М. Мова опису складних зображень // Матеріали 4-ї української конференції «Автоматика 97».- Черкаси , 1997. - Т. III. - С. 95.

81. Березька К.М. Опис, аналіз та синтез складних зображень // Інформаційні технології і системи. - 1998. - Т.1, №1/2. - С.168-173.

82. Зыков А.А. Основы теории графов. - М.: Наука, 1987. - 384 с.

83. Павлидис Т. Алгоритмы машинной графики и обработки изображений. - М.: Радио и связь, 1986. - 398 с.

84. Роджерс Д., Адамс Дж. Математические основы машинной графики: Пер. с англ./ Пер. Ю.П.Кулябичев, В.Г.Иваненко, ред. Ю.И.Топчеев. - М.: Машиностроение, 1980. - 240 с.

85. Роджерс Д. Алгоритмические основы машинной графики: Пер. с англ. - М.: Мир, 1989. - 512 с.

86. Фоли Дж., Вэн Дэм А. Основы интерактивной машинной графики: В 2-х книгах: Пер с англ. - М.: Мир, 1985. - Кн.1. - 367 с.

87. Беллман Р. Введение в теорию матриц: Пер с англ. - М.: Наука, 1976. - 352 с.

88. Березька К.М. Формалізація груп перетворень // Матеріали Всеукраїнської наукової конференції «Застосування обчислювальної техніки, математичного моделювання та математичних методів у наукових дослідженнях». - Львів, 1999. - С. 6-7.

89. Березька К.М. Алгоритми побудови груп перетворень // Вісник Львівського державного університету. Прикладна математика та інформатика. - 1999. - Вип.1. - С. 8-12.

90. Глушков В.М., Капитонова Ю.В., Мищенко А.Т. Логическое проектирование дискретных устройств. - Киев: Наук. думка, 1987. - 264 с.

91. Биркгоф Г., Барти Т. Современная прикладная алгебра: Пер. с англ. - М.: Мир, 1976. - 400 с.

92. Єрьоміна Н.В. Проектування баз даних: Навч. посібник. - К.: КНЕУ, 1998. - 208с.

93. Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем. - М.: Финансы и статистика, 1989. - 351 с.

94. Диго С.М. Проектирование и использование баз данных: Учебник Пер. с англ. - М.: Финансы и статистика, 1995. - 208 с.

95. Кузнецов С.Д. Ведення в СУБД // СУБД. - 1996. - № 1-3.

96. Системы управления базами данных и знаний / Под ред. А.Н. Наумова. - М.: Финансы и статистика, 1991. - 352 с.

97. Мартинес Ф. Синтез изображений. Принципы, аппаратное и программное обеспечение: Пер с франц. - М.: Радио и связь, 1990. - 192 с.

98. Ситник В. Ф., Орленко Н.С. Імітаційне моделювання: Навч. посібник. - К.: КНЕУ, 1998. - 232 с.

99. Д.Кнут. Искусство программирования для ЭВМ: В 7 т.: Пер с англ. - М.: Мир,

1977. - Т.2: Получисленные алгоритмы. - 724 с.

100. Мартин Ф. Моделирование на вычислительных машинах: Пер. с англ. - М.: Изд-во «Советское радио», 1972. - 288 с.

101. Вентцель Е.С. Теория вероятностей. - М.: Наука, 1964. - 578 с.

102. Гмурман В.Е. Теория вероятностей и математическая статистика. - М.: Высшая школа, 1972. - 368 с.

103.Сурков Д.А., Сурков К.А., Вальвачев А.Н. Программирование в среде Borland Pascal для Windows: Справочное пособие. - Минск: Вышэйшая школа, 1996. - 432с.

104. Вальвачев А.Н., Крисевич В.С. Программирование на языке ПАСКАЛЬ для персональных ЭВМ ЕС: Справочное пособие. - Минск: Вышэйшая школа, 1989. - 223 с.

105. Йорг Шиб. Windows 95: сотни полезных рецептов: Пер с нем. - К.: BHV, 1996. - 560 с.

106. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++: Пер. с англ. - М.: «Издательство Бином», СПб: «Невский диалект», 1998. - 560 с.

107. Шлеер С., Меллор С. Объектно-ориентированный анализ: моделирование мира в состояниях: Пер. с англ. - К.: Диалектика, 1993. - 240 с.

108. Сташин В.В., Уросов А.Ф., Мологонцев О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах. - М.: Энергоатомиздат, 1990. - 224 с.

109. Каган Б.М., Сташин В.В. Основы проектирования микропроцессорных устройств автоматики. - М.: Энергоатомиздат, 1987. - 304 с.

110. Сопряжение датчиков и устройств ввода данных с компьютера IBM PC: Пер. с англ. / Под ред. У. Томпкина, Дж. Уэбстера. - М.: Мир, 1992. - 592 с.

111. Степаненко О.С. Информационно-вычислительные системы на основе I486. - К.: УКСП «Кобза», 1994. - 216 с.

112. СверхБИС универсальных однокристальных микро-ЭВМ / Кобылинский А.В.,

Липовецкий Г.П., Сабадаш Н.Г. и др. - К.: Техніка, 1987. - 166 с.

Додаток А

Приклади вишивок орнаментальних груп на полосі

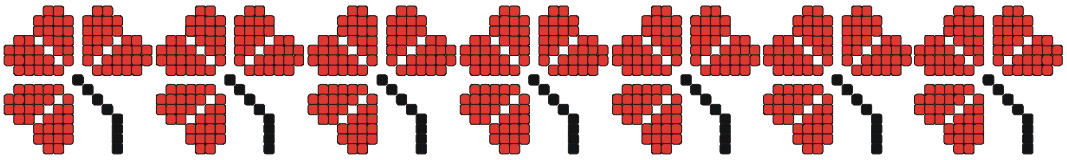


Рис. А.1.Фрагмент вишивки групи p1.

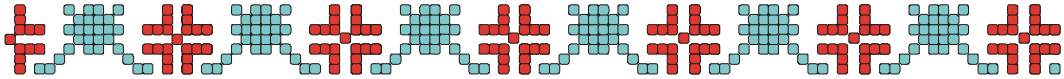


Рис. А.2.Фрагмент вишивки групи p1m.

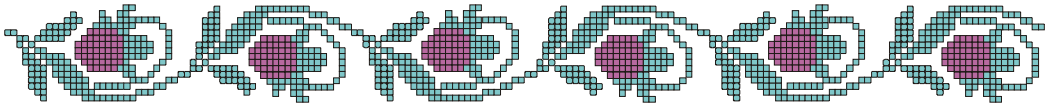


Рис. А.3.Фрагмент вишивки групи pg.

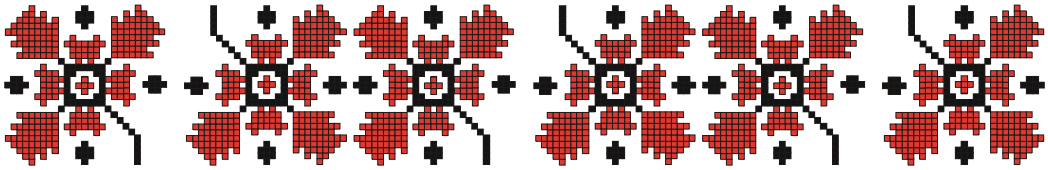


Рис. А.4.Фрагмент вишивки групи p2.

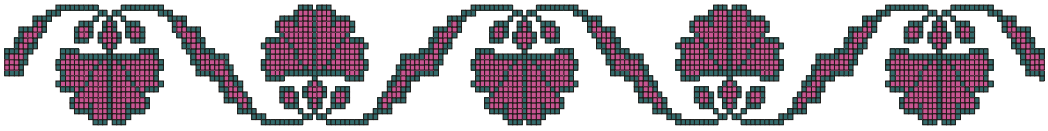


Рис. А.5.Фрагмент вишивки групи pmg.

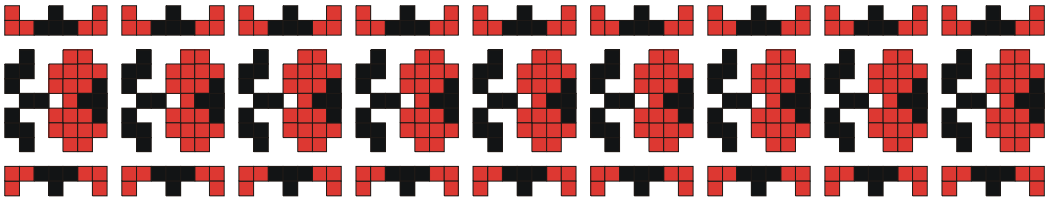


Рис. А.6.Фрагмент вишивки групи pm.

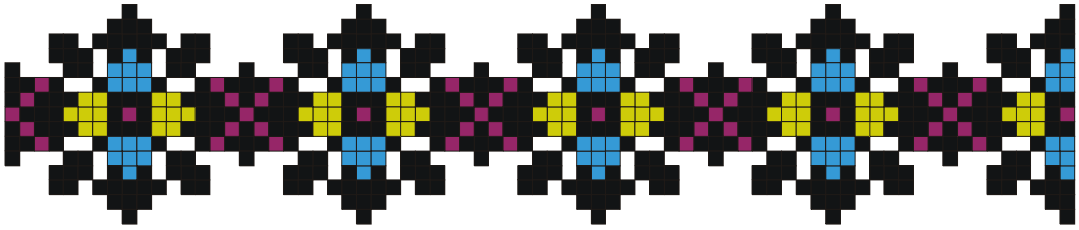


Рис. А.6.Фрагмент вишивки групи pmm.

Додаток Б

Приклади вишивок орнаментальних груп на площині

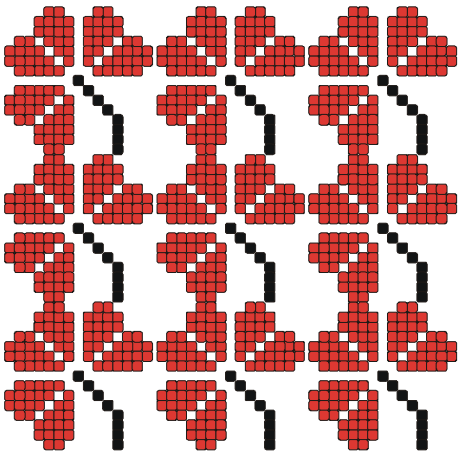


Рис. Б.1.Фрагмент вишивки групи p1.

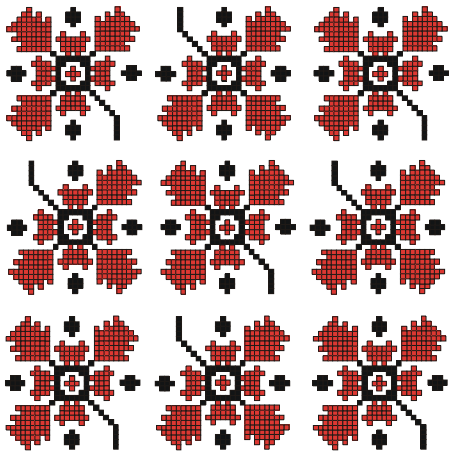


Рис. Б.2.Фрагмент вишивки групи p2.

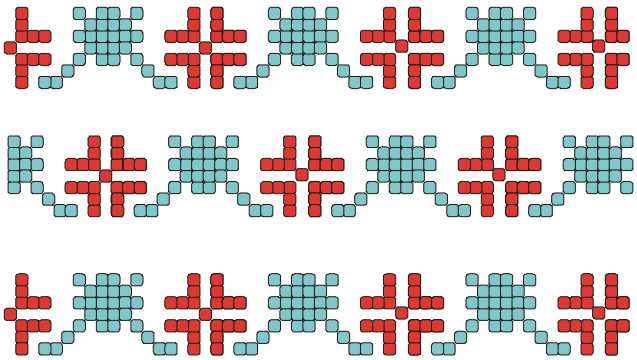


Рис. Б.3.Фрагмент вишивки групи pm.

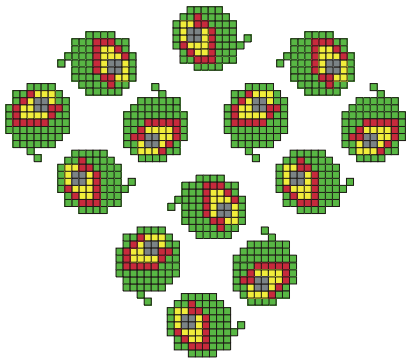


Рис. Б.4.Фрагмент вишивки групи p4.

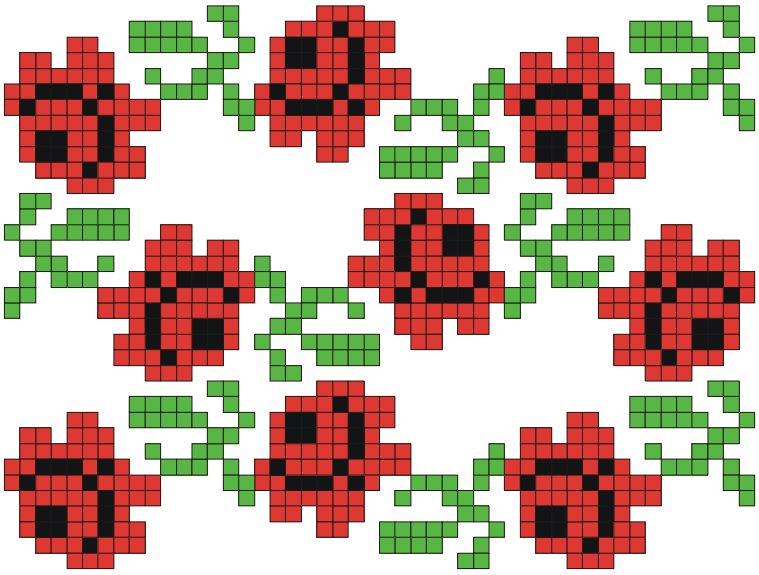


Рис. Б.5.Фрагмент вишивки групи pgg.

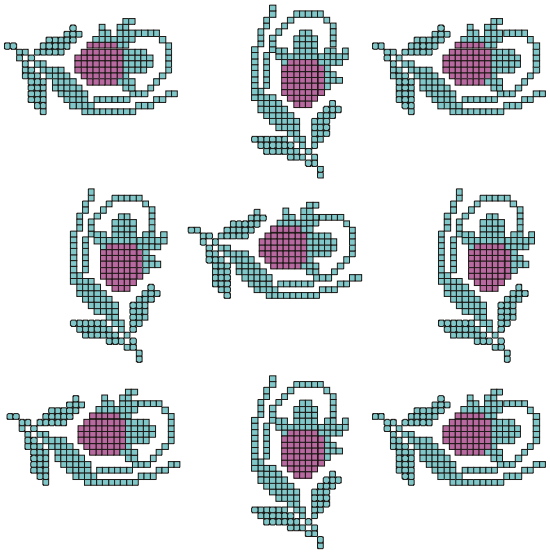


Рис. Б.6.Фрагмент вишивки групи pg.

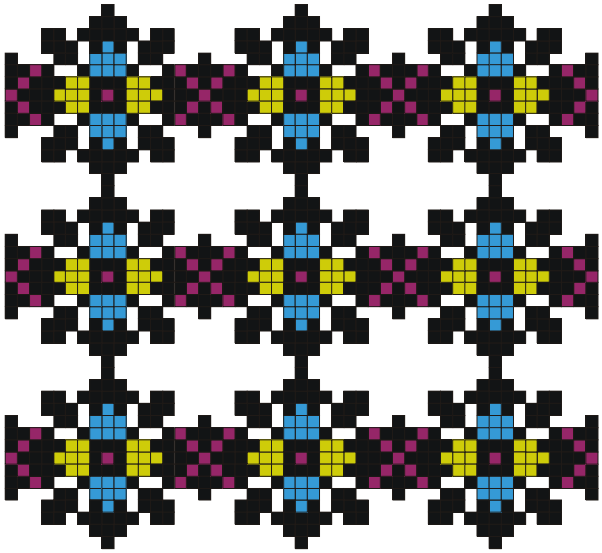


Рис. Б.7.Фрагмент вишивки групи pmm.

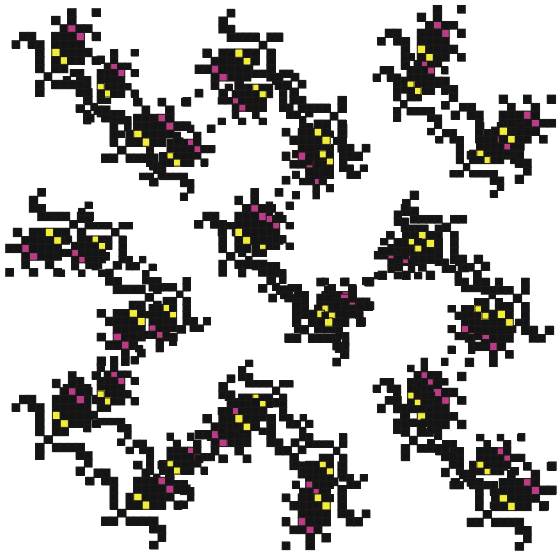


Рис. Б.8.Фрагмент вишивки групи pmg.

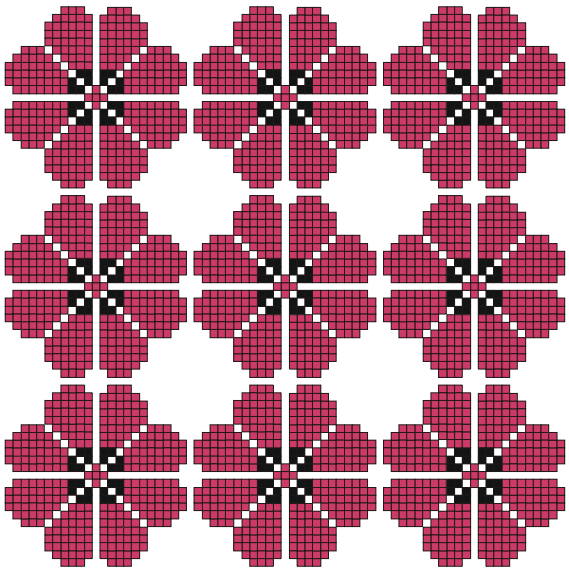


Рис. Б.9.Фрагмент вишивки групи p4m.

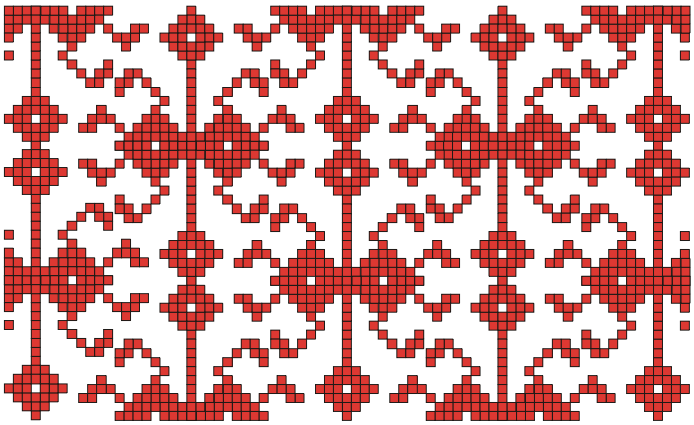


Рис. Б.10.Фрагмент вишивки групи cmm.

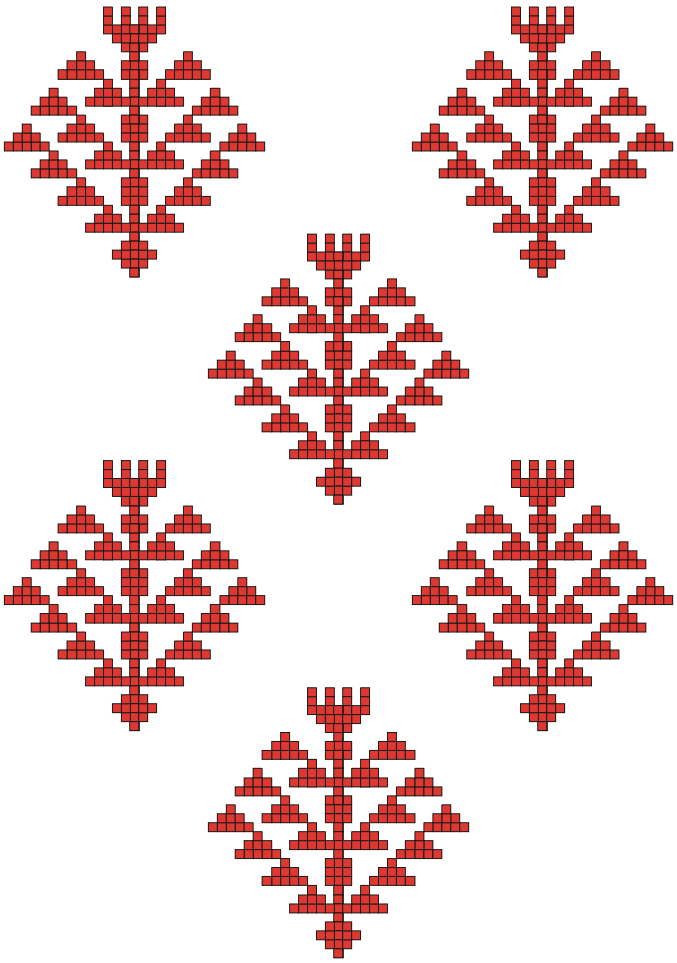


Рис. Б.11.Фрагмент вишивки групи cm.

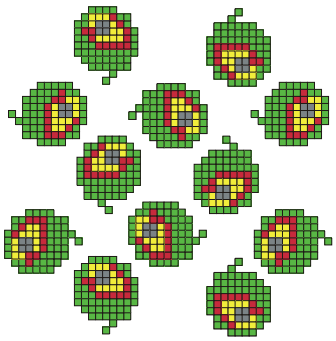


Рис. Б.12.Фрагмент вишивки групи p4g.

Додаток Д
Формули мінімальних рисунків орнаментів

Рис. В.1.а.

c(8,10) + (~b(8,10))+ (~d(2,10))+ b(7,10) + (~c(7,10))+ d(4,10) + c(4,10) + (~b(4,10))+ +(~d(2,10))+ b(3,10) + (~c(3,10))+ (~d(6,10))+ a(2,10) + c(8,1) + (~b(9,1))+ d(2,1) + +b(10,1) + (~c(8,1))+ a(5,1) + c(5,1) + a(2,1) + b(5,1) + (~a(2,1))+ (~b(4,1))+ (~a(2,1))+ +(~c(4,1))+ a(13,1) + (~b(8,1))+ c(8,1) + b(4,1) + (~d(2,1))+ (~b(4,1))+ (~c(7,1))+ +b(8,1) + a(5,1) + c(7,1) + (~d(2,1))+ (~c(6,1))+ (~a(2,1))+ (~b(3,1))+ (~b(6,1))+ +d(2,1)+ c(3,1) + b(4,1) + c(6,1) + (~d(2,1))+ (~c(6,1))+ (~b(4,1))+ (~c(2,1))+ +(~d(2,1))+ b(5,1) + c(8,1) + (~d(2,1)) + (~c(7,1))+ c(8,1) + (~b(12,1))+ d(2,1) + +b(12,1)+ (~a(8,1))+ c(2,1) + d(2,1) + (~b(6,1))+ (~c(10,1))+ a(2,1) + c(9,1) + b(5,1) + +d(2,1) + a(8,1) + (~b(10,1))+ d(2,1) + c(7,1) + a(2,1) + (~c(7,1))+ (~d(2,1))+ b(9,1) + +d(6,1) + (~b(4,1))+ d(2,1) + c(4,1) + (~d(2,1))+ (~c(3,1))+ (~d(2,1))+ b(3,1) + +(~b(9,1))+ a(2,1) + c(2,1) + (~a(2,1))+(~c(5,1))+(~b(5,1))+ (~a(2,1))+ b(6,1) + c(4,1) + +(~c(6,1))+ (~c(7,1))+ (~b(4,1))+ (~c(3,1))+ b(6,1) + c(8,1) + d(2,1) + (~c(8,1))+ +(~b(6,1))+ c(3,1) + b(4,1) + c(7,1) + (~b(7,1))+ (~a(2,1))+ b(7,1) + (~a(4,1))+ +(~d(2,1))+ (~c(3,1))+ (~b(4,1))+ (~c(8,1))+ c(2,1) + (~d(2,1))+ b(8,1) + a(2,1) + +(~b(9,1))+ d(2,1) + c(7,1) + b(4,1) + c(3,1) + (~b(6,1))+ (~c(9,1))+ (~d(2,1))+ c(9,1) + +b(5,1) + d(6,1) + (~b(2,0))+ (~a(6,1))+ (~c(7,1))+ d(2,1) + c(6,1) + (~a(4,1))+ +(~c(3,10))+ d(3,10) + a(2,10) + a(19,10) + a(1,10) + (~d(14,0))+ (~a(19,0))+ a(1,10) + +@(1,18,10)+ @(11,7,10) + @(23,6,10) + @(30,24,10) + @(19,23,10) + @(15,27,10) + +@(24,29,10) + a(26,0) + (~c(3,1))+ (~d(2,16))+ b(2,16) + (~d(2,16))+ (~c(2,16))+ +(~b(11,10))+ d(4,16) + a(2,16) + a(4,0) + a(1,10) + a(9,0) + d(2,0) + (~d(3,16))+ +d(5,16) + (~c(2,16))+@(30,13,1) + @(29,2,3)+@(25,25,3) + @(19,29,3) + @(10,1,3).

Рис. В.1.б.

(~c(3,1))+ nb(3,0) + c(3,1) + b(3,1) + (~c(3,1))+ na(5,0) + (~b(3,1))+ c(3,1) + b(3,1) + +nc(3,0) + (~b(3,1))+ (~c(3,1))+ nd(4,0) + na(4,0) + c(2,1) + b(3,1) + nd(3,0) + +(~na(3,0))+ (~b(3,1))+ (~nc(6,0))+ (~na(3,0))+ a(1,1) + nd(6,0) + (~na(4,0))+ c(5,1).

Рис. В.1.в.

a(2,3) + a(3,12) + a(3,10) + a(2,1) + (~b(7,1))+ c(2,1) + a(1,10) + b(3,10) + a(1,10) + +b(3,10) + a(1,10) + b(3,10) + a(1,10) + (~a(4,1))+ (~a(2,1))+ d(2,1) + (~b(2,1))+ +(~a(2,1))+ d(2,1) +(~b(2,1))+(~d(2,1))+ (~d(3,10))+ (~d(2,12))+ @(0,1,10) + d(8,10) + +a(3,10)+ a(1,10) + b(3,10) + a(1,10) + b(3,10) + a(1,10) + b(3,10) + a(1,10) + c(2,10) + +(~b(7,1))+ a(3,1) + a(3,10) + a(3,12) + a(1,3) + (~d(2,3))+ (~d(3,12))+ (~d(3,10))+ +(~b(5,1))+ a(3,1) + a(2,10) + (~d(2,10))+ (~a(2,10))+ b(2,10) + a(1,1) + (~a(9,0))+ +d(3,0) + a(1,16) + (~d(7,0))+ (~d(2,12))+ @(1,1,10).

Рис.В.1.г.

(~d(2,1))+(~c(45,1))+d(5,0)+d(2,1)+c(41,1) + (~d(2,1))+ (~c(40,1))+ (~d(3,1))+a(2,1) + +d(2,1)+c(3,0)+c(2,1)+(~d(2,1))+(~b(2,1))+c(4,0)+(~d(2,1))+a(2,1)+d(2,1)+c(3,0)+a(2,1)+ (~b(2,1)) +a(2,1) +c(3,0)+a(2,1)+(~b(2,1))+a(2,1)+c(3,0)+a(2,1)+(~b(2,1))+a(2,1)+ +c(3,0)+ a(2,1)+(~b(2,1))+a(2,1) + c(3,0) + a(2,1) + (~b(2,1))+ a(2,1) + c(3,0) + a(2,1) + +(~b(2,1))+a(2,1)+c(3,0)+a(2,1)+(~b(2,1))+a(2,1) + c(3,0) + a(2,1) + (~b(2,1))+ a(2,1) +

+c(3,0)+a(2,1)+(~b(2,1))+ a(2,1) + c(3,0) + a(2,1) + (~b(2,1))+ a(2,1) + c(3,0) + a(2,1) +
+c(3,1) + (~a(3,0))+ (~c(2,1))+ a(2,1) + (~a(8,0))+ c(3,1) + (~a(3,1))+ (~c(21,1))+ b(2,1) +
c(20,1) + d(2,1) + (~c(20,1))+ (~b(8,1))+ c(5,1) + d(2,1) + (~c(6,1))+ b(7,1) + c(2,1) +
+(~b(6,1))+c(5,1)+b(3,1) + (~c(3,1))+ b(3,1) + c(5,1) + (~b(8,1))+ (~c(13,1))+ d(2,1) +
+c(13,1) + b(9,1) + (~c(6,1)) + (~b(4,1)) + c(3,1) + (~a(2,1)) + (~c(3,1)) +b(5,1)+c(7,1)+
+(~b(10,1))+(~c(13,1))+d(3,0)+c(13,1)+b(11,1)+ a(2,1) + (~b(11,1))+ a(2,1) + b(10,1) +
+c(2,0) + c(9,1) + (~a(2,1))+ (~c(8,1))+ (~b(8,1))+ (~a(2,1))+ b(8,1) + c(8,1) +
+(~a(13,0))+b(6,1)+d(6,0)+ a(3,12) + (~c(2,12))+ (~d(2,0))+ c(3,4) + (~a(5,0))+ b(2,4)+ +
@(23,45,7) + (~a(14,0))+ (~c(11,1))+ d(2,1) + c(11,1) + a(2,1) + (~a(4,0))+ (~a(5,1))+
+b(2,1) + a(3,1) + (~c(2,1))+ (~a(3,1))+ (~c(5,1))+ (~d(3,1))+ c(6,1) + (~a(2,1))+
+(~c(4,1))+ d(4,0) + (~b(2,1))+ d(3,1) + a(3,1) + (~c(2,1))+ b(2,1) + (~d(2,1))+
+(~b(2,1))+@(1,44,12)+@(1,44,11) + (~d(18,0))+ c(5,1) + b(9,1) + (~c(14,1))+ d(2,1) +
+c(13,1)+(~b(8,1))+(~c(6,1))+d(2,1)+b(2,1)+c(5,1)+ b(7,1)+(~c(12,1))+d(5,0)+ d(3,1) +
+b(3,1) + (~c(2,1))+ a(2,1) + (~b(2,1))+ c(3,0) + c(6,1) + a(2,1) + (~c(7,1))+ (~b(4,1))+
+d(2,1) + b(3,1) + a(2,1) + c(6,1) + (~b(4,1))+ (~c(4,1))+ d(2,1) + b(2,1) + c(3,1) +
+b(3,1) + (~a(2,1))+ (~b(2,1))+ (~c(3,1))+ (~d(4,0))+ (~a(2,0))+ (~a(2,1))+ (~b(2,1))+
+d(2,1) + b(2,1) + (~d(22,0))+ a(4,0) + c(6,1) + b(6,1) + (~a(2,1))+ (~b(5,1))+ +(~c(5,1))+
a(2,1) + c(4,1) + b(4,1) + a(5,1) + c(46,1) + d(2,1) + (~a(2,1))+ (~c(4,1))+ +(~b(4,1))+
(~a(5,1))+ (~c(4,1))+ b(4,1) + c(3,1) + b(3,1) + a(2,1) + (~b(4,1))+ +(~c(3,1))+ (~b(3,1))+
c(3,1) + a(2,1) + (~c(3,1))+ b(2,1) + c(3,1) + b(4,1) + c(2,0) + +(~b(4,1))+a(2,1)+ b(4,1) +
c(4,1) + (~c(7,0))+ (~c(5,1))+ (~a(2,1))+ c(5,1) + (~a(7,0))+ +b(3,1) + (~d(2,1))+
(~b(5,1))+ d(2,1) + b(2,1) + (~a(3,1))+ (~c(5,1))+ b(4,1) + c(3,1) + +b(3,1) + a(2,1) +
c(2,1) + (~a(3,1))+ (~b(3,1))+ (~c(3,1))+ (~b(3,1))+ c(4,1) + a(2,1) + +b(4,1) + (~c(2,1))+
(~b(3,1))+ (~c(3,1))+ (~b(2,1))+ c(3,1) + (~c(6,0))+ (~b(2,1))+ +(~c(6,1))+ b(4,1) + c(3,1)
+ b(3,1) + a(2,1) + c(7,1) + (~b(2,1))+ (~d(2,1))+ (~c(4,1))+ +(~b(4,1))+(~d(2,1))+b(3,1)+
(~d(2,1))+ (~b(5,1))+ d(2,1) + a(1,1) + (~c(5,1))+ b(3,1) + +c(3,1) + b(3,1) + a(2,1) +
(~b(4,1))+ (~c(3,1))+ (~b(2,1))+ c(4,1) + a(2,1) + (~d(8,0))+ +(~c(4,1))+ (~b(6,1))+
(~c(6,1))+ b(4,1) + c(2,1) + c(2,1) + b(3,1) + a(2,1) + (~b(4,1))+
+(~c(3,1))+(~b(3,1))+c(5,1) + b(6,1) + c(5,1) + (~c(6,1))+ d(2,1) + (~b(5,1))+ (~c(4,1))+
+b(2,1) + c(3,1) + b(4,1) + (~c(6,1))+ (~b(6,1))+ (~c(6,1))+ b(4,1) + c(3,1) + b(3,1) +
+a(2,1) + c(7,1) + (~c(5,1))+ (~b(6,1))+ (~c(5,1))+ b(3,1) + c(3,1) + b(3,1) + a(2,1) +
+(~b(4,1))+ (~c(3,1))+ (~b(2,1))+ c(4,1) + b(5,1) + (~c(7,1))+ (~b(6,1))+ (~c(6,1))+
+b(4,1) + c(3,1) + b(3,1) + a(2,1) + c(7,1) + (~c(11,1))+ (~a(2,1))+ c(6,1) + (~b(4,1))+
+(~c(3,1))+ (~b(2,1))+ c(4,1) + b(5,1) + d(2,1) + (~b(5,1))+ (~c(5,1))+ b(3,1) + c(3,1) +
+b(3,1) + (~d(5,0))+ (~c(3,14))+ c(45,14) + (~d(2,14))+ (~c(44,14))+ a(2,14) + c(43,1)+ +
(~d(2,1))+ (~c(42,1))+ a(2,1) + c(41,3) + (~d(2,3))+ (~c(40,1))+ a(2,1) + c(39,1) +
+(~d(2,1))+ (~c(38,1))+ c(6,1) + b(6,1) + a(5,1) + c(24,1) + d(2,1) + (~c(24,1))+
+(~a(2,1))+ c(25,1) + d(3,0) + (~c(27,0))+ (~b(5,1))+ (~a(2,1))+ b(5,1) + d(5,0) +
+(~b(2,1))+ c(4,1) + b(4,1) + (~c(4,1))+ (~b(2,1))+ a(2,1) + (~b(3,1))+ c(3,1) + b(3,1) +
+(~c(2,1))+ d(7,0) + b(5,1) + a(2,1) + (~b(6,1))+ a(2,1) + b(5,1) + c(20,1) + d(3,1) +
+(~c(20,1))+ (~b(4,1))+ c(23,1) + (~d(3,1))+ (~c(21,1))+ b(2,1) + (~d(2,1))+ (~b(3,1))+
+c(21,1) + (~d(7,0))+ (~c(18,1))+ (~c(11,0))+ (~a(3,0))+ (~b(3,1))+ (~c(3,1))+ a(2,1) +
+c(2,1) + b(2,1) + a(10,0) + a(2,0) + c(22,1) + (~d(2,1))+ (~c(21,1))+ a(2,1) + c(20,1) +
+d(4,0) + (~c(23,11))+ a(5,0) + c(4,10) + b(4,10) + (~a(2,10))+ (~b(3,10))+ (~c(3,10))+

+@(46,0,1) + a(7,0) + (~b(4,1))+ d(2,1) + b(5,1) + a(2,1) + (~b(5,1))+ a(4,0) + b(2,1) + +c(3,1) + (~b(3,1))+ (~c(3,1))+ a(2,1) + b(2,1) + c(2,1) + (~b(2,1))+ d(5,0) + b(6,1) + +(~c(5,1))+ (~d(2,1))+ a(1,1) + c(6,1) + (~b(6,1))+ a(2,1) + b(6,1) + (~c(6,1))+ a(2,0) + +c(7,11) + (~d(3,0))+ (~c(5,1))+ (~a(2,1))+ c(6,1) + (~d(3,1))+ (~c(4,1))+ d(9,0) + +b(3,1) + c(2,1) + (~b(6,1))+ (~c(2,1))+ b(3,1) + a(2,1) + b(3,1) + (~b(6,1))+ a(3,1) + +c(5,1)+ d(2,1) + (~c(5,1))+ b(2,1) + c(4,1) + (~d(2,1))+ (~c(4,11))+ b(4,11) + d(2,11) + +(~b(3,11))+ c(3,11) + @(52,17,10) + d(14,0) + (~c(19,10))+ (~b(6,10))+ c(24,10) + +@(61,32,10)+ @(36,11,12) + @(53,7,7) + @(29,3,14) + @(42,46,10) + @(33,36,10) + +@(24,27,10)+ @(15,18,10) + @(6,9,10) + @(9,12,12) + @(18,21,12) + @(27,30,12) + +@(36,39,12)+@(39,42,14)+ @(30,33,14) + @(21,24,14) + @(12,15,14) + @(3,6,14) + +@(0,3,12)+ (~b(11,0))+ (~a(19,0))+ (~c(10,10))+ (~b(10,10))+ (~a(7,0))+ (~c(10,10))+ +(~d(5,0))+c(13,12)+ b(11,12) + (~c(8,12))+ (~b(6,12))+ c(4,12) + (~c(11,0))+ c(3,10) + +b(6,10) + (~c(8,10))+ (~b(4,10))+ @(0,14,12) + @(24,43,11).

Рис.В.1.д.

(~d(3,12))+ a(1,12) + a(2,12) + a(3,8) + (~b(3,8))+ +(~d(2,8))+ (~na(3,0))+ a(1,8) + +(~d(2,3))+ nc(3,0) +c(8,8) + c(2,8) + a(1,7) + c(2,7) + a(1,12) + (~d(2,12))+ na(2,0) + +nd(3,0) + a(1,12) + (~d(2,12)) + (~d(5,8))+ (~b(3,8))+ a(2,8) + (~c(2,8))+ b(3,1) + +(~d(4,1))+ a(1,8) + (~c(2,8))+ (~d(3,8))+ a(2,8) +d(3,12) + (~b(2,12))+ (~c(4,1))+ +(~a(4,1))+ (~b(3,1))+c(4,1) + c(2,8) + c(3,1) + (~nd(4,0))+ a(1,8) +(~nc(4,0))+ a(1,8) + +nd(3,0) + a(2,1) + d(3,1) + a(2,1)+@(9,6,12) + @(9,6,12) + @(9,6,8) + (~nd(7,0))+ +na(3,0) + (~a(2,3)) .

Рис. В.1.е.

(~d(2,1))+ (~c(2,1))+ (~d(2,1))+ (~c(2,1))+ (~a(2,1))+ d(2,1) + c(2,1) + @(2,1,10) + +(~nb(2,0))+ a(1,10).

Рис. В.1.є

c(2,10) + (~d(2,12))+ na(3,10) + d(4,10) + nc(3,10) + (~d(6,1))+ na(3,10) + d(5,10) + +a(5,10) + (~d(5,10))+ c(2,10) + a(3,10) + d(7,1) + a(4,1) + (~d(8,1))+ (~na(4,1))+ +a(1,1) + (~nb(8,1))+ c(11,1) + (~b(2,1))+ b(3,1) + nd(3,1) + a(1,1) + (~nd(7,1))+ +(~b(2,10))+ (~a(4,10))+ (~nc(3,0))+ a(4,12) + b(2,12) + (~nc(3,0))+ (~b(2,11))+ +(~a(4,11))+ (~nc(3,0))+ a(4,10) + b(2,10) + (~na(7,0))+ a(1,1) + nb(4,0) + +(~a(2,13))+ (~d(2,13))+ a(3,13) + d(2,13) + (~nd(3,0))+ (~a(3,12))+ (~d(2,12))+ +a(3,12) + na(3,0) + a(2,7) + na(3,0) + a(2,12) + (~b(2,12))+ a(2,12) + d(2,12) + +(~a(2,10))+ d(2,10) + a(2,10) + d(2,10) + (~a(2,13))+ d(2,13) + a(2,13) + d(2,13) + +(~a(2,11)) .

Рис.В.2.а.

(~d(2,10))+ b(2,10) + a(2,10) + c(2,10) + d(5,10) + (~c(3,10))+ @(4,2,10) + (~na(3,0))+ +(~a(2,10))+(~b(2,10))+ d(2,10) + c(2,10) + a(5,10) + (~c(3,10))+ @(2,4,10) + nb(5,0)+ + na(3,0) + a(2,10) + c(2,10) + d(2,10) + (~b(4,10))+ (~d(5,10))+ @(9,1,10) + na(5,0) + +nd(3,0) + a(2,10) + d(4,10) + (~a(5,10))+ b(3,10) + a(2,10) + d(2,10) + (~a(2,10))+ +(~nb(6,0))+ d(4,10) + (~a(4,10))+ (~d(2,10))+ b(4,10) + @(5,8,10) + (~c(2,10))+ +(~a(5,10))+ (~b(2,10))+ d(2,10) + c(2,10) + a(2,10) + b(3,10) + @(2,9,10) + nb(3,0) + +na(2,0) + c(4,1) + c(2,1) + d(3,1).

Рис.В.2.б.

приведено в 3.2

Рис.В.2.в

c(2,1) + d(3,1) + b(2,1) + a(3,1) + c(2,1) + d(2,1) + (~b(2,1))+ (~b(8,1))+ (~d(11,1))+ +a(9,0) + d(5,0) + a(3,1) + c(2,1) +

(~b(2,1))+ c(2,1) + d(2,1) + (~b(2,1))+ d(2,1) + +(~a(4,1))+ (~c(2,1))+ (~a(5,1))+ (~a(2,1))+ d(3,0) + c(2,1) + d(2,1)+(~b(2,1))+ d(2,1) + +(~b(2,1))+ c(2,1) + a(3,1) + b(3,1) + (~d(3,1))+ b(2,1) + a(3,1) + d(3,1) + (~b(2,1))+ +(~a(3,1))+ d(4,0) + (~a(2,1))+ a(8,1) + (~d(2,1))+ a(3,1) + (~d(2,1))+ a(4,1) + +(~d(2,1))+ a(5,1) + (~d(2,1))+ a(3,1) + (~d(8,0))+ (~b(2,1))+ (~a(2,1))+ d(2,1) + +(~b(2,1))+ (~a(2,1))+ d(2,1) + (~b(2,1))+ (~a(4,1))+ d(5,1) + @(15,18,7)+@(6,18,14)+ +@(0,17,14) + @(3,8,14) + @(7,11,14) + (~d(18,0))+ (~a(6,1))+ b(4,1) + a(8,1) + +(~b(4,1))+ (~a(2,1))+ (~a(7,1))+ (~c(3,0))+ (~c(2,1))+ (~a(3,1))+ @(11,2,7) .

Рис.В.2.г

d(2,10) + c(2,10) + b(2,10) + d(2,10) + c(2,10) + b(2,10) + d(7,10) + (~a(7,10))+ b(2,10) + (~c(2,10))+ (~a(2,10))+ b(2,10) + (~c(2,10))+ d(11,0) + a(3,1) + (~c(2,1))+ d(3,1) + +d(6,0) + a(7,10) + d(7,10) + (~c(2,10))+ (~b(2,10))+ d(2,10) + (~c(2,10))+ (~b(2,10))+ +d(2,10) + (~c(5,10))+ a(2,10) + b(2,10) + (~c(2,10))+ a(2,10) + b(2,10) + a(7,0) + +(~d(3,0))+ b(2,1)+(~d(9,1))+(~c(2,1))+ c(4,1) + (~d(3,1))+ (~a(2,1))+ a(8,1) + b(2,1) + +(~b(3,1))+ (~a(2,1))+ d(2,1) + a(3,1) + (~d(2,1))+ d(8,1) + c(7,1) + d(5,1) + (~a(8,0))+ +(~d(11,0))+ (~d(3,1))+ (~a(2,1))+ d(3,1) + (~a(5,1))+ (~d(3,1))+ (~a(2,1))+ d(3,1) + +(~d(7,0))+ a(1,1) + d(3,0) + a(2,0) + b(3,10) + c(3,10) + (~b(3,10))+ (~c(2,10))+ +(~a(5,0))+ (~a(2,10))+ d(2,10) + (~a(3,10))+ d(2,10) + (~c(2,10))+ b(2,10) + +(~d(5,10))+ (~b(2,10))+ a(4,10) + d(3,10) + b(2,10) + b(5,0) + (~c(2,10))+ (~d(4,10))+ +(~b(2,10))+ a(7,10)+(~c(2,10))+d(4,10) + (~b(2,10))+ (~d(2,10))+ (~a(3,10))+ a(8,0) + +a(7,10) + (~c(2,10))+ b(2,10) + a(2,10) + (~c(2,10))+ b(2,10) + a(2,10) + (~c(5,10))+ +d(2,10) + (~b(2,10))+ (~c(2,10))+ d(2,10) + (~b(2,10))+ (~c(2,10))+ d(7,10) + d(5,0) + +(~b(2,10)) + a(2,10) +d(3,10)+(~a(2,10))+c(2,10)+a(4,10)+(~b(2,10))+(~d(7,10))+ +c(2,10) + (~a(3,10))+ c(3,0) + a(3,0) + a(3,1) + (~c(2,1))+ d(3,1) + (~c(13,0))+ +(~d(3,1))+ (~b(2,1))+ a(3,1) + d(26,0) + (~a(3,1))+ c(2,1) + (~d(3,1))+ (~d(7,0))+ +(~a(3,0))+ b(2,10) + d(2,10) + a(4,10) + (~c(2,10))+ d(4,0) + a(3,10) + (~b(2,10))+ +(~d(3,10))+ (~a(5,0))+ d(3,10) + (~c(2,10))+ a(4,10) + (~a(6,0))+ (~d(3,16))+ d(9,0) + +a(9,0) + a(1,16) + @(15,21,10) + @(6,21,10) + @(14,14,10) + @(7,14,10) + +@(21,14,10) + @(14,7,10) + @(22,6,10) + @(4,5,10) .

Рис. В.2.д

в 3.2

Рис. В.2.е

c(4,3) + a(2,3) + a(2,3) + d(2,3) + a(8,3) + b(2,3) + (~d(3,10))+ b(2,10) + a(3,10) + +c(2,10) + d(3,10) + (~b(2,10))+ (~a(3,10))+ (~d(4,10))+ a(2,10) + c(2,10) + d(2,10) + +(~a(2,10))+ nd(4,10) + a(3,10) + c(2,10) + d(3,10) + (~b(2,10))+ (~a(3,10))+ +(~c(2,10))+ (~d(3,10))+ a(3,10) + c(2,10) + d(2,10) + (~a(3,10))+ (~d(2,10))+ +(~nb(3,10))+ (~a(4,3))+ (~c(2,3))+ b(2,3) + (~d(2,3))+ c(2,3) + (~d(3,3))+ c(2,3) + +(~na(6,0))+ c(2,3) + (~c(4,3))+ (~nd(3,0))+ b(2,3) + (~d(2,3))+ c(2,3) + (~d(3,3))+ +c(2,3) + nd(8,0) + c(8,3) + b(2,3) + (~d(2,3))+ a(3,3) + (~c(2,3))+ d(3,3) + nd(4,0) + +a(2,3) + d(2,3) + (~c(3,3))+ a(2,3).

Рис. В.2.є

(~d(3,10))+ nc(3,0) + (~a(3,10))+ nd(3,0) + a(3,10) + (~nb(3,0))+ (~d(3,10))+ nc(3,0) + +a(2,1) + b(4,1) + (~c(2,1))+ a(2,1) + (~d(2,1))+ (~c(2,1))+ na(3,0) + d(5,1).

Рис. В.2.ж

nd(8,0) + b(4,10) + a(3,10) + (~b(3,10))+ (~a(2,10))+ b(2,10) + (~d(3,10))+ b(4,10) + +c(8,10) + (~nc(2,10))+ (~a(5,10))+ (~d(2,10))+ a(2,10) + (~c(2,10))+ (~d(2,10))+ +nd(6,0) + a(5,10) + (~b(6,10))+ (~na(3,0))+ (~c(3,10))+ (~d(2,10))+ (~c(3,10))+ +a(3,10) + c(3,10) + (~a(2,10))+ (~c(2,10))+ na(6,0) + a(2,10) + (~b(2,10))+ d(2,10).

Рис. В.2.з

d(2,10) + a(2,10) + (~d(3,10))+ a(5,10) + d(3,10) + a(10,10) + (~d(6,10))+ d(4,10) + +(~a(2,10))+d(2,10)+(~a(2,10))+ (~a(8,0))+ b(2,10) + (~d(2,10))+ (~a(8,10))+ d(2,10) + +a(2,10) + (~d(4,10))+ a(2,10) + d(2,10) + a(3,0) + (~d(2,10))+ a(2,10) + d(2,10) + +@(6,4,1) + b(2,0) + a(2,0) + a(8,1) + d(2,1) + (~b(3,1))+ (~a(4,1))+ (~d(3,1))+ a(4,0) + +d(2,1) + b(2,1) + @(12,1,10) .

Рис. В.2.и

+ (~b(4,1))+ (~a(2,1))+ b(4,11) + (~a(2,11))+ (~b(4,11))+ (~a(2,11))+ b(5,1) + a(2,1) + (~a(3,1))+ (~a(7,14))+ d(2,14) + c(3,14) + a(2,14) + b(3,14) + (~a(3,14))+ d(2,14) + (~a(2,14))+ (~d(2,12))+ (~a(2,12))+ (~na(3,0))+ (~d(2,1))+ d(4,1) + c(2,1) + a(3,1) + @(1,4,1) + na(6,0) + b(3,12) + d(2,12) + (~b(2,12)) .

Рис. В.3.а.

(~d(3,1))+ b(3,1) + a(3,1) + c(4,1) + b(2,1) + (~d(2,1))+ c(2,1) + (~a(2,1))+ d(3,0) + +c(6,1) + a(2,1) + b(6,1) + d(3,1) + (~b(5,1))+ c(5,1) + d(3,1) + (~c(3,1))+ (~a(3,1))+ +d(3,1) + (~a(2,1))+ (~d(3,0))+ (~a(7,1))+ d(3,1) + (~a(2,1))+ (~d(3,0))+ (~a(6,1))+ +(~c(3,1))+(~d(3,1))+b(4,1) + (~d(3,1))+ (~c(2,1))+ (~a(2,1))+ a(8,0) + c(3,1) + b(8,1) + +a(2,1) + d(2,1) + (~b(7,1))+ (~a(2,0))+ b(7,12) + a(2,0) + d(7,0) + (~b(8,10))+ +(~d(3,10))+ (~a(7,11))+ (~d(3,11))+ a(8,11) + (~c(2,11))+ (~a(9,13))+ b(2,13) + +a(7,13) + (~c(2,13))+ (~a(6,13))+ @(5,4,10) + d(4,0) + d(3,10) + (~a(2,10))+ +(~d(3,10))+ @(2,13,12) + @(8,11,11) + @(15,13,10) + @(19,15,12).

Рис. В.3.б

(~c(2,1))+ (~b(2,1))+ d(7,1) + (~a(3,1))+ (~b(2,1))+ c(3,1) +a(2,1)+ c(2,1) + a(3,1) + +d(3,1)+(~a(2,1))+ (~d(3,0))+a(5,1) + d(3,1) + (~a(2,1))+ (~d(3,0))+ a(5,1)+ (~d(3,1)) + +(~a(4,1))+ (~d(3,1))+ a(4,1) + (~d(4,0))+ (~b(2,1))+(~a(3,1))+ (~c(3,1))+ (~d(4,1))+ +a(3,0) + d(2,0) + d(3,1) +a(3,1) +(~c(2,1))+ b(2,1)+ d(6,0)+ (~a(3,0))+ +(~c(3,1))+(~b(3,1))+ (~d(5,1))+ (~a(2,1))+ a(1,10) + d(7,0) + a(4,10) +(~c(2,10))+ +(~a(2,10)) + d(3,10) + a(2,10) + @(8,7,1) +@(5,5,1).

Рис. В.3.в

(~a(3,1))+ b(3,1) + d(3,0) + a(2,1) + c(3,1) + d(4,1) + c(2,1) + d(3,1) + a(8,1) + c(5,1) + +d(5,1) + (~b(2,1))+ (~a(2,1))+ (~c(2,1))+ (~a(2,1))+ (~d(3,1))+ a(2,1) + (~a(5,1))+ +d(3,1) + (~b(2,1))+ a(4,0) + a(1,1) + (~a(7,0))+ b(2,1) + (~d(3,1))+ a(3,1) + (~a(5,1))+ +(~c(2,1))+ (~a(5,1))+ (~c(3,1))+ (~d(6,1))+ b(2,1) + a(2,1) + b(2,1) + (~d(2,1))+ +(~a(4,1))+ (~d(2,1))+ b(2,1) + a(9,0) + d(7,1) + a(2,1) + b(2,1) + a(2,1) + (~d(3,1))+ +a(2,1) + (~d(8,1))+ (~b(4,1))+ b(2,1) + (~c(2,1))+ a(4,0) + a(6,1) + c(2,1) + b(3,0) + +(~c(2,1))+ (~a(7,1))+ d(5,0) + a(4,1) + c(2,1) + d(3,0) + (~a(2,0))+ (~c(2,1))+ +(~a(4,1))+ c(7,0) + (~b(2,1))+ (~a(4,0))+ d(5,11) + (~a(2,11))+ (~d(5,11))+ (~a(4,0))+ +d(5,13) + (~a(2,13))+ (~d(5,13))+ (~a(5,0))+ (~d(3,7))+ (~a(3,7))+ d(4,7) + c(3,7) + +(~d(3,7))+ @(6,14,7) + (~a(4,0))+ (~c(3,13))+ (~d(3,13))+ a(3,13) + d(4,13) + +(~d(8,0))+(~d(2,10))+ (~a(2,10))+ (~b(2,10))+ a(2,10) + a(3,0) + (~d(2,12))+ a(2,12) + +d(2,12)+a(2,12)+ d(2,12) + d(4,4) + @(4,9,4) + @(5,13,7) + @(2,12,13) + @(2,15,8) + +@(8,15,14) + @(12,7,14) + @(17,17,12) + @(13,15,10).

Рис. В.3.г

a(3,10) + b(2,10) + c(2,10) + b(2,10) + c(2,10) + b(2,10) + (~d(2,10))+ (~c(2,10))+ +b(2,10) + (~d(3,10))+ (~c(2,10))+ (~a(3,10))+ (~b(2,10))+ (~c(2,10))+ (~d(2,10))+ +(~b(2,10))+ d(4,10) + c(2,10) + (~b(2,10))+ b(3,10) + a(2,1) + d(2,1) + d(8,1) + +(~b(2,10))+ (~c(2,10))+ (~d(2,10))+ (~a(3,10))+ d(2,10) + c(2,10) + (~b(2,10))+ +d(3,10) + a(2,10) + c(3,10) + a(2,10) + b(2,10) + (~c(2,10))+ (~a(2,10))+ d(2,10) + +a(2,10) + (~d(3,10))+ a(4,10) + (~d(2,10))+ (~a(3,10))+ (~d(2,10))+ b(2,10) + +(~d(2,10))+ (~a(2,10))+ d(2,10) + (~b(2,10))+ (~a(4,10))+ c(2,10) + d(2,10) + +(~a(2,10))+ (~d(2,10))+ a(6,10) + d(3,10) + c(4,1) + (~d(5,1))+ b(2,1) + (~d(3,1))+ +b(4,1) + a(3,10) + (~d(2,10))+ (~a(3,10))+ (~d(2,10))+ a(2,10) + (~c(2,10))+ +(~a(4,10))+ (~a(2,1))+ (~c(2,1))+ (~d(3,1))+ c(2,1) + d(2,1) + a(2,1) + (~d(3,1))+ +a(2,1)+ d(7,1) + (~a(6,1))+ a(2,1) + d(2,1) + c(2,1) + d(4,1) + a(3,1) + d(7,1) + d(2,1) + +(~b(2,1))+ (~a(5,1))+ (~b(3,1))+ d(3,1) + b(2,1) + (~d(2,1))+ b(2,1) + d(2,1) + a(2,1) + +(~d(2,1))+ a(3,1) + a(3,1) + a(2,1) + (~d(2,1))+ (~d(4,1))+ (~d(3,1))+ c(2,1) + +(~d(4,1))+ c(2,1) + (~d(4,1))+ a(2,1) + d(6,1) + (~b(2,1))+ d(5,1) + (~b(3,1))+ +(~a(2,1))+ (~b(2,1))+ d(4,1) + b(4,1) + d(2,1) + a(4,1) + a(2,1)

+ (~c(3,1))+ d(2,1) + +(~c(2,1))+ b(2,1) + (~d(7,1))+ a(2,1) + b(3,10) + c(2,10) + d(3,10) + (~a(2,10))+ +(~d(3,10))+ d(5,10) + (~a(2,1))+ c(3,1) + d(3,1) + c(3,1) + d(2,1) + (~b(2,1))+ +(~a(3,1))+ (~d(2,1))+ a(4,1) + (~d(5,1))+ a(3,1) + b(4,1) + (~c(4,1))+ (~a(3,1))+ +(~b(3,1))+ a(4,1) + a(1,12) + a(1,12) + a(5,12) + d(3,12) + a(2,1) + d(2,1) + b(2,1) + +a(2,1) + b(2,1) + a(3,1) + (~d(2,1))+ a(5,1) + c(2,1) + a(6,1) + c(2,1) + a(3,1) + c(2,1) + +a(3,1) + c(2,1) + a(4,1) + (~a(8,1))+ (~d(2,1))+ (~a(5,1))+ b(2,1) + (~a(6,1))+ +(~c(2,1))+(~a(4,1))+(~b(2,1))+(~a(2,1))+ d(3,1) + a(2,1) + (~d(2,1))+ a(3,1) + d(3,1) + +a(3,1) + (~d(3,1))+ c(2,1) + a(5,1) + d(4,1) + (~a(5,1))+ (~d(2,1))+ a(11,1) + b(2,1) + +(~a(7,1))+ (~a(13,1))+ (~d(4,7))+ (~b(2,7))+ d(5,7) + (~a(2,7))+ (~d(5,7))+ (~b(2,7))+ +d(3,7)+(~a(2,7))+(~d(3,7))+ (~b(2,7))+ d(2,7) + (~a(4,7))+ c(2,7) + (~a(2,7))+ d(7,7) + +(~c(2,7))+ (~d(4,7))+ c(3,7) + d(4,7) + c(2,7) + (~d(4,7))+ c(2,7) + d(4,7) + c(2,7) + +(~d(4,7))+c(2,7) + d(3,7) + a(2,7) + d(2,7) + c(2,7) + (~d(3,7))+ b(2,7) + d(4,7) + +c(2,7) + (~d(4,7))+ d(3,7) + a(2,7) + d(3,7) + a(3,1) + d(2,1) + b(2,1) + (~d(5,1))+ +(~c(2,1))+ (~d(4,1))+ (~a(2,1))+ (~c(5,1))+ (~a(2,1))+ (~c(2,1))+ (~a(4,1))+ d(3,1) + +a(2,1) + d(2,1) + c(4,1) + d(2,1) + a(2,1) + (~d(2,1))+ c(6,1) + d(2,1) + (~a(2,1))+ +(~d(16,1))+ (~a(4,1))+ d(4,7) + c(6,7) + d(2,7) + a(6,7) + d(2,7) + a(5,7) + d(2,7) + +a(3,7) + d(2,7) + b(2,7) + a(3,7) + d(2,7) + a(7,7) + (~c(4,7))+ (~a(3,7))+ (~d(3,7))+ +a(4,7)+b(3,7)+a(2,7)+ (~a(19,7))+ (~d(2,7))+ (~a(3,7))+ (~d(3,7))+ (~a(2,7))+ d(3,7) + +(~c(5,7))+d(7,7)+a(6,7)+ a(3,1) + c(2,1) + a(3,1) + c(2,1) + a(2,1) + a(5,1) + (~d(2,7))+ +a(3,1) + d(4,1) + (~a(2,1))+ (~c(2,1))+ (~a(3,1))+ (~a(11,1))+ d(2,1) + a(3,1) + c(2,1) + +a(3,1) + d(2,1) + (~b(2,1))+ (~a(2,1))+ a(4,1) + a(4,1) + (~c(2,1))+ (~a(3,1))+ c(3,1) + +a(5,1)+(~d(2,1))+d(3,1)+a(2,1) + d(2,1) + a(3,1) + c(2,1) + a(6,1) + (~a(4,1))+ d(5,1) + +(~b(3,1))+ (~a(2,1))+ (~d(3,1))+ b(3,1) + d(4,1) + c(2,1) + a(2,1) + c(2,1) + (~a(2,1))+ +(~a(4,1))+ (~d(4,1))+ b(5,1) + (~a(2,1))+ (~d(2,1))+ (~c(2,1))+ (~a(3,1))+ (~c(3,1))+ +a(3,1)+ c(2,1)+(~a(2,1))+ d(5,1) + (~a(2,1))+ (~a(2,1))+ (~b(3,1))+ d(3,1) + (~b(2,1))+ +(~a(3,1))+ d(2,1) + (~c(2,1))+ (~d(2,1))+ b(3,1) + a(2,1) + (~a(3,1))+ (~d(2,1))+ +(~a(2,7))+ (~d(2,7))+ (~c(2,7))+ d(2,7) + (~a(3,7))+ (~d(3,7))+ c(2,7) + (~c(3,7))+ +(~a(4,7))+ d(3,7) + a(4,7) + (~d(2,7))+ (~a(4,7))+ d(3,7) + d(3,1) + (~a(2,1))+ d(3,1) + +(~a(2,1))+(~d(2,1))+(~a(8,1))+(~d(5,1))+ d(4,1) + c(2,1) + (~d(3,1))+ c(2,1) + d(4,1) + +a(2,1)+d(2,1)+(~a(4,1))+(~a(2,1))+(~d(2,1))+ (~a(2,1))+ a(2,1) + (~d(3,1))+ (~a(4,1))+ +(~b(2,1))+(~d(4,1))+ (~a(4,1))+ (~b(2,1))+ (~a(3,1))+ (~b(2,1))+ (~a(2,1))+ (~b(2,1))+ +(~a(2,1))+(~b(2,1))+ (~a(2,1))+ (~b(2,1))+ a(5,10) + d(2,10) + (~b(2,10))+ (~d(2,10))+ +(~a(7,10))+c(2,10) + a(2,10) + (~d(3,10))+ (~a(3,10))+ b(2,10) + a(3,10) + (~d(2,10))+ +(~a(4,10))+ (~d(2,10))+ a(4,10) +(~d(2,10))+ (~a(3,10))+ b(2,10) + a(3,10) + c(2,10) + +b(2,10) + c(2,10) + d(3,10) + (~b(2,10))+ (~d(4,10))+ (~a(3,10))+ d(2,10) + c(2,10) + +(~d(8,0))+ a(4,10) + c(2,10) + (~a(3,10))+ (~a(4,0))+ (~c(2,10))+ (~d(2,10))+ a(2,10) + +d(2,10)+ b(3,10) + (~a(2,10))+ (~d(5,10))+ (~a(3,10))+ d(2,10) + c(2,10) + (~d(2,10))+ +a(3,10) + (~d(4,10))+ a(2,10) + c(2,10) + d(2,10) + (~c(2,10))+ c(4,0) + a(3,10) + +(~d(2,10))+(~a(2,10))+(~b(3,10))+d(3,10)+c(2,10) + (~d(3,10))+ (~a(3,10))+ d(2,10) + +(~d(5,10))+ (~b(3,1))+ a(2,1) + d(2,1) + @(19,22,1) + @(31,28,7) + @(42,32,7) + +@(47,33,7) + @(49,35,7) + @(28,32,1) + @(44,46,1) + @(5,3,10) + @(10,26,1) + +c(5,0) + a(5,1) + (~d(2,1))+ a(7,1) + c(7,0) + (~b(3,1))+ @(3,14,1) .

Додаток Ж

Ж.1. Процедура створення орнаментної групи

```
procedure Raport( xn, yn : Integer );
var BegMemID, i, j : Integer ;
begin
  if Step > 9999 then
    begin
      ShowMessage( ' забагато заповнень : не вистачає пам`яті ' );
      Brk := true ;
    end;
  IterForm.Left := ( ZoomC*PArray(xn,yn).x )div 4 + MainForm.PBox.Left + MainForm.Scroll.Left + MainForm.Left + 5 ;
  IterForm.Top  := ( ZoomC*PArray(xn,yn).y )div 4 + MainForm.PBox.Top  + MainForm.Scroll.Top  + MainForm.Top + 42
;
  IterForm.ShowModal ;
  if not Brk then
    begin
      BegMemID := ElNo ;
      ElemNo := 1 ;
      Brk := false ;
      FMove := false ;
      ElemNo := ElemNo + 1 ;
      With MainForm do
        begin
          Screen.Cursor := crHourGlass ;
          UndoBtn.Enabled := true ;
          PUndo.Enabled := true ;
          ClearBtn.Enabled := true ;
          PClear.Enabled := true ;
          ViewBtn.Enabled := true ;
          PView.Enabled   := true ;
          PrntBtn.Enabled := true ;
          PPrnt.Enabled   := true ;
          FrmlBtn.Enabled := true ;
          PFrml.Enabled   := true ;
          SaveBtn.Enabled := true ;
          PSave.Enabled   := true ;
          MoveBtn.Enabled := true ;
          Gauge1.BorderStyle := bsSingle ;
          Gauge1.BackColor := clWhite ;
          Gauge1.ShowText := true ;
          case RapNo of
          0 : begin
              ProgBit := 50 ;
              VirtRot ;
              for i := -Iter to Iter do
              for j := -Iter to Iter do
                Elem( xn+i, yn+j, CurElem, CurElNm );
             end;
          1 : begin
               case NNet.kind  of
                 rect : begin
                          ProgBit := 20 ;
                          VirtRot ;
                          for i := -Iter to Iter do
                          for j := -Iter to Iter do
                            begin
```

```
                Elem( xn + i*3    , yn + j*3    , CurElem, CurElNm );
                Elem( xn + i*3 + 1, yn + j*3    , CurElem, CurElNm );
                Elem( xn + i*3 - 1, yn + j*3    , CurElem, CurElNm );
                Elem( xn + i*3    , yn + j*3 + 1, CurElem, CurElNm );
                Elem( xn + i*3    , yn + j*3 - 1, CurElem, CurElNm );
              end;
            end;
      gect :  begin
              ProgBit := 13 ;
              if odd(xn) then
               begin
                 VirtRot ;
                 Elem( xn    , yn    , CurElem, CurElNm );
                 Elem( xn + 1, yn    , CurElem, CurElNm );
                 Elem( xn - 1, yn    , CurElem, CurElNm );
                 Elem( xn - 1, yn + 1, CurElem, CurElNm );
                 Elem( xn + 1, yn + 1, CurElem, CurElNm );
                 Elem( xn    , yn + 1, CurElem, CurElNm );
                 Elem( xn    , yn - 1, CurElem, CurElNm );
               end
              else
               begin
                 VirtRot ;
                 Elem( xn    , yn    , CurElem, CurElNm );
                 Elem( xn + 1, yn    , CurElem, CurElNm );
                 Elem( xn - 1, yn    , CurElem, CurElNm );
                 Elem( xn - 1, yn - 1, CurElem, CurElNm );
                 Elem( xn + 1, yn - 1, CurElem, CurElNm );
                 Elem( xn    , yn + 1, CurElem, CurElNm );
                 Elem( xn    , yn - 1, CurElem, CurElNm );
               end;
            end;
    end;
  end;
2 : begin
    ProgBit := 10 ;
    VirtRot ;
    for i := -Iter to Iter do
    for j := -Iter to Iter do
     begin
       Elem( xn + i*6    , yn + j*6    , CurElem, CurElNm );
       Elem( xn + i*6 - 2, yn + j*6    , CurElem, CurElNm );
       Elem( xn + i*6    , yn + j*6 - 2, CurElem, CurElNm );
       Elem( xn + i*6 + 2, yn + j*6    , CurElem, CurElNm );
       Elem( xn + i*6    , yn + j*6 + 2, CurElem, CurElNm );
     end;
    Rotate( Pi, false );
    for i := -Iter to Iter do
    for j := -Iter to Iter do
     begin
       Elem( xn + i*6     , yn + j*6    , RotElem, RotElNm );
       Elem( xn + i*6 - 2 , yn + j*6    , RotElem, RotElNm );
       Elem( xn + i*6     , yn + j*6 - 2, RotElem, RotElNm );
       Elem( xn + i*6 + 2 , yn + j*6    , RotElem, RotElNm );
       Elem( xn + i*6     , yn + j*6 + 2, RotElem, RotElNm );
     end;
  end;
3 : begin
    ProgBit := 10 ;
    VirtRot ;
    for i := -Iter to Iter do
```

```
      for j := -Iter to Iter do
       begin
        Elem( xn + i*6     , yn + j*3     , CurElem, CurElNm );
        Elem( xn + i*6 - 2 , yn + j*3     , CurElem, CurElNm );
        Elem( xn + i*6     , yn + j*3 - 1 , CurElem, CurElNm );
        Elem( xn + i*6 + 2 , yn + j*3     , CurElem, CurElNm );
        Elem( xn + i*6     , yn + j*3 + 1 , CurElem, CurElNm );
       end;
      Rotate( 0, true );
      for i := -Iter to Iter do
      for j := -Iter to Iter do
       begin
        Elem( xn + i*6     , yn + j*3     , RotElem, RotElNm );
        Elem( xn + i*6 - 2 , yn + j*3     , RotElem, RotElNm );
        Elem( xn + i*6     , yn + j*3 - 1 , RotElem, RotElNm );
        Elem( xn + i*6 + 2 , yn + j*3     , RotElem, RotElNm );
        Elem( xn + i*6     , yn + j*3 + 1 , RotElem, RotElNm );
       end;
     end;
4 : begin
      ProgBit := 5 ;
      VirtRot ;
      for i := -Iter to Iter do
      for j := -Iter to Iter do
       begin
        Elem( xn + i*6     , yn + j*6     , CurElem, CurElNm );
        Elem( xn + i*6 - 2 , yn + j*6     , CurElem, CurElNm );
        Elem( xn + i*6     , yn + j*6 - 2 , CurElem, CurElNm );
        Elem( xn + i*6 + 2 , yn + j*6     , CurElem, CurElNm );
        Elem( xn + i*6     , yn + j*6 + 2 , CurElem, CurElNm );
        Elem( xn + i*6 - 1 , yn + j*6 - 1 , CurElem, CurElNm );
        Elem( xn + i*6 - 3 , yn + j*6 - 1 , CurElem, CurElNm );
        Elem( xn + i*6 - 1 , yn + j*6 - 3 , CurElem, CurElNm );
        Elem( xn + i*6 + 1 , yn + j*6 - 1 , CurElem, CurElNm );
        Elem( xn + i*6 - 1 , yn + j*6 + 1 , CurElem, CurElNm );
       end;
      Rotate( -Pi/2, true );
      for i := -Iter to Iter do
      for j := -Iter to Iter do
       begin
        Elem( xn + i*6 + 1 , yn + j*6     , RotElem, RotElNm );
        Elem( xn + i*6     , yn + j*6 + 1 , RotElem, RotElNm );
        Elem( xn + i*6 - 1 , yn + j*6     , RotElem, RotElNm );
        Elem( xn + i*6 - 2 , yn + j*6 + 1 , RotElem, RotElNm );
        Elem( xn + i*6 + 1 , yn + j*6 - 2 , RotElem, RotElNm );
        Elem( xn + i*6     , yn + j*6 - 1 , RotElem, RotElNm );
        Elem( xn + i*6 + 3 , yn + j*6     , RotElem, RotElNm );
        Elem( xn + i*6 + 2 , yn + j*6 + 1 , RotElem, RotElNm );
        Elem( xn + i*6 + 1 , yn + j*6 + 2 , RotElem, RotElNm );
        Elem( xn + i*6     , yn + j*6 + 3 , RotElem, RotElNm );
       end;
     end;
5 : begin
      ProgBit := 10 ;
      VirtRot ;
      for i := -Iter to Iter do
      for j := -Iter to Iter do
       begin
        Elem( xn + i*3     , yn + j*3     , CurElem, CurElNm );
        Elem( xn + i*3 - 1 , yn + j*3     , CurElem, CurElNm );
        Elem( xn + i*3     , yn + j*3 - 1 , CurElem, CurElNm );
```

```
        Elem( xn + i*3 + 1 , yn + j*3     , CurElem, CurElNm );
        Elem( xn + i*3 , yn + j*3 + 1     , CurElem, CurElNm );
      end;
    Rotate( -Pi/2, true );
    for i := -Iter to Iter do
    for j := -Iter to Iter do
      begin
        Elem( xn + i*3 + 1 , yn + j*3 + 1 , RotElem, RotElNm );
        Elem( xn + i*3 + 1 , yn + j*3 + 2 , RotElem, RotElNm );
        Elem( xn + i*3     , yn + j*3 + 1 , RotElem, RotElNm );
        Elem( xn + i*3 + 2 , yn + j*3 + 1 , RotElem, RotElNm );
        Elem( xn + i*3 + 1 , yn + j*3     , RotElem, RotElNm );
      end;
  end;
6 : begin
    ProgBit := 5 ;
    VirtRot ;
    for i := -Iter to Iter do
    for j := -Iter to Iter do
      begin
        Elem( xn + i*6 - 1 , yn + j*6     , CurElem, CurElNm );
        Elem( xn + i*6 - 3 , yn + j*6     , CurElem, CurElNm );
        Elem( xn + i*6 - 1 , yn + j*6 - 2 , CurElem, CurElNm );
        Elem( xn + i*6 + 1 , yn + j*6     , CurElem, CurElNm );
        Elem( xn + i*6 - 1 , yn + j*6 + 2 , CurElem, CurElNm );
      end;
    Rotate( 0, true );
    for i := -Iter to Iter do
    for j := -Iter to Iter do
      begin
        Elem( xn + i*6 + 1 , yn + j*6     , RotElem, RotElNm );
        Elem( xn + i*6 - 1 , yn + j*6     , RotElem, RotElNm );
        Elem( xn + i*6 + 1 , yn + j*6 - 2 , RotElem, RotElNm );
        Elem( xn + i*6 + 3 , yn + j*6     , RotElem, RotElNm );
        Elem( xn + i*6 + 1 , yn + j*6 + 2 , RotElem, RotElNm );
      end;
    Rotate( Pi, false );
    for i := -Iter to Iter do
    for j := -Iter to Iter do
      begin
        Elem( xn + i*6 + 1 , yn + j*6     , RotElem, RotElNm );
        Elem( xn + i*6 - 1 , yn + j*6     , RotElem, RotElNm );
        Elem( xn + i*6 + 1 , yn + j*6 - 2 , RotElem, RotElNm );
        Elem( xn + i*6 + 3 , yn + j*6     , RotElem, RotElNm );
        Elem( xn + i*6 + 1 , yn + j*6 + 2 , RotElem, RotElNm );
      end;
    Rotate( Pi, true  );
    for i := -Iter to Iter do
    for j := -Iter to Iter do
      begin
        Elem( xn + i*6 - 1 , yn + j*6     , RotElem, RotElNm );
        Elem( xn + i*6 - 3 , yn + j*6     , RotElem, RotElNm );
        Elem( xn + i*6 - 1 , yn + j*6 - 2 , RotElem, RotElNm );
        Elem( xn + i*6 + 1 , yn + j*6     , RotElem, RotElNm );
        Elem( xn + i*6 - 1 , yn + j*6 + 2 , RotElem, RotElNm );
      end;
  end;
7 : begin
    ProgBit := 5 ;
    VirtRot ;
    Elem( xn     , yn     , CurElem, CurElNm );
```

```
    Elem( xn - 1 , yn     , CurElem, CurElNm );
    Elem( xn - 1 , yn - 2 , CurElem, CurElNm );
    Elem( xn + 1 , yn     , CurElem, CurElNm );
    Elem( xn + 1 , yn + 2 , CurElem, CurElNm );
    Rotate( Pi, false );
    Elem( xn + 1 , yn + 1 , RotElem, RotElNm );
    Elem( xn     , yn + 1 , RotElem, RotElNm );
    Elem( xn     , yn - 1 , RotElem, RotElNm );
    Elem( xn + 2 , yn + 1 , RotElem, RotElNm );
    Elem( xn + 2 , yn + 3 , RotElem, RotElNm );
    Rotate( 0 , true  );
    Elem( xn + 1 , yn + 1 , RotElem, RotElNm );
    Elem( xn     , yn + 1 , RotElem, RotElNm );
    Elem( xn     , yn - 1 , RotElem, RotElNm );
    Elem( xn + 2 , yn + 1 , RotElem, RotElNm );
    Elem( xn + 2 , yn + 3 , RotElem, RotElNm );
    Rotate( Pi, true );
    Elem ( xn + 1 , yn + 2 , RotElem, RotElNm );
    Elem ( xn     , yn + 2 , RotElem, RotElNm );
    Elem ( xn     , yn     , RotElem, RotElNm );
    Elem ( xn + 2 , yn + 2 , RotElem, RotElNm );
    Elem ( xn + 2 , yn + 4 , RotElem, RotElNm );
  end;
8 : begin
    ProgBit := 5 ;
    VirtRot ;
    for i := -Iter to Iter do
    for j := -Iter to Iter do
     begin
       Elem( xn + i*6 - 1 , yn + j*6     , CurElem, CurElNm );
       Elem( xn + i*6 - 3 , yn + j*6     , CurElem, CurElNm );
       Elem( xn + i*6 - 1 , yn + j*6 - 2 , CurElem, CurElNm );
       Elem( xn + i*6 + 1 , yn + j*6     , CurElem, CurElNm );
       Elem( xn + i*6 - 1 , yn + j*6 + 2 , CurElem, CurElNm );
     end;
    Rotate( 0, true );
    for i := -Iter to Iter do
    for j := -Iter to Iter do
     begin
       Elem( xn + i*6     , yn + j*6 - 1 , RotElem, RotElNm );
       Elem( xn + i*6 - 2 , yn + j*6 - 1 , RotElem, RotElNm );
       Elem( xn + i*6     , yn + j*6 - 3 , RotElem, RotElNm );
       Elem( xn + i*6 + 2 , yn + j*6 - 1 , RotElem, RotElNm );
       Elem( xn + i*6     , yn + j*6 + 1 , RotElem, RotElNm );
     end;
    Rotate( Pi, false );
    for i := -Iter to Iter do
    for j := -Iter to Iter do
     begin
       Elem( xn + i*6 + 1 , yn + j*6     , RotElem, RotElNm );
       Elem( xn + i*6 - 1 , yn + j*6     , RotElem, RotElNm );
       Elem( xn + i*6 + 1 , yn + j*6 - 2 , RotElem, RotElNm );
       Elem( xn + i*6 + 3 , yn + j*6     , RotElem, RotElNm );
       Elem( xn + i*6 + 1 , yn + j*6 + 2 , RotElem, RotElNm );
     end;
    Rotate( Pi, true  );
    for i := -Iter to Iter do
    for j := -Iter to Iter do
     begin
       Elem( xn + i*6     , yn + j*6 + 1 , RotElem, RotElNm );
       Elem( xn + i*6 - 2 , yn + j*6 + 1 , RotElem, RotElNm );
```

```
          Elem( xn + i*6     , yn + j*6 - 1 , RotElem, RotElNm );
          Elem( xn + i*6 + 2 , yn + j*6 + 1 , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 + 3 , RotElem, RotElNm );
        end;
    end;
 9 : begin
      ProgBit := 5 ;
      VirtRot ;
      for i := -Iter to Iter do
      for j := -Iter to Iter do
        begin
          Elem( xn + i*6     , yn + j*6     , CurElem, CurElNm );
          Elem( xn + i*6 - 2 , yn + j*6     , CurElem, CurElNm );
          Elem( xn + i*6     , yn + j*6 - 2 , CurElem, CurElNm );
          Elem( xn + i*6 + 2 , yn + j*6     , CurElem, CurElNm );
          Elem( xn + i*6     , yn + j*6 + 2 , CurElem, CurElNm );
        end;
      Rotate( -Pi/2 , true );
      for i := -Iter to Iter do
      for j := -Iter to Iter do
        begin
          Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
          Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
        end;
      Rotate( Pi/2 , true );
      for i := -Iter to Iter do
      for j := -Iter to Iter do
        begin
          Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
          Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
        end;
      Rotate( Pi , false );
      for i := -Iter to Iter do
      for j := -Iter to Iter do
        begin
          Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
          Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
        end;
    end;
10 : begin
      ProgBit := 5 ;
      VirtRot ;
      for i := -Iter to Iter do
      for j := -Iter to Iter do
        begin
          Elem( xn + i*6     , yn + j*6     , CurElem, CurElNm );
          Elem( xn + i*6 - 2 , yn + j*6     , CurElem, CurElNm );
          Elem( xn + i*6     , yn + j*6 - 2 , CurElem, CurElNm );
          Elem( xn + i*6 + 2 , yn + j*6     , CurElem, CurElNm );
          Elem( xn + i*6     , yn + j*6 + 2 , CurElem, CurElNm );
        end;
      Rotate( Pi/2 , false);
```

```
      for i := -Iter to Iter do
      for j := -Iter to Iter do
       begin
         Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
         Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
       end;
      Rotate( Pi , false);
      for i := -Iter to Iter do
      for j := -Iter to Iter do
       begin
         Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
         Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
       end;
      Rotate( -Pi/2 , false);
      for i := -Iter to Iter do
      for j := -Iter to Iter do
       begin
         Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
         Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
       end;
     end;
11 : begin
      ProgBit := 2 ;
      VirtRot ;
      for i := -Iter to Iter do
      for j := -Iter to Iter do
       begin
         Elem( xn + i*6     , yn + j*6     , CurElem, CurElNm );
         Elem( xn + i*6 - 2 , yn + j*6     , CurElem, CurElNm );
         Elem( xn + i*6     , yn + j*6 - 2 , CurElem, CurElNm );
         Elem( xn + i*6 + 2 , yn + j*6     , CurElem, CurElNm );
         Elem( xn + i*6     , yn + j*6 + 2 , CurElem, CurElNm );
       end;
      Rotate( Pi/2 , true );
      for i := -Iter to Iter do
      for j := -Iter to Iter do
       begin
         Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
         Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
       end;
      Rotate( Pi , true );
      for i := -Iter to Iter do
      for j := -Iter to Iter do
       begin
         Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
         Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
         Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
```

```
        end;
      Rotate( Pi/2 , false);
      for i := -Iter to Iter do
      for j := -Iter to Iter do
        begin
          Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
          Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
        end;
      Rotate( 0 , true );
      for i := -Iter to Iter do
      for j := -Iter to Iter do
        begin
          Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
          Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
        end;
      Rotate( -Pi/2 , false);
      for i := -Iter to Iter do
      for j := -Iter to Iter do
        begin
          Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
          Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
        end;
      Rotate( Pi , false );
      for i := -Iter to Iter do
      for j := -Iter to Iter do
        begin
          Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
          Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
        end;
      Rotate( -Pi/2 , true );
      for i := -Iter to Iter do
      for j := -Iter to Iter do
        begin
          Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
          Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
          Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
        end;
    end;
12 : begin
      ProgBit := 2 ;
      VirtRot ;
      for i := -Iter to Iter do
      for j := -Iter to Iter do
        begin
          Elem( xn + i*6     , yn + j*6     ,  CurElem, CurElNm );
          Elem( xn + i*6 - 2 , yn + j*6     ,  CurElem, CurElNm );
          Elem( xn + i*6     , yn + j*6 - 2 ,  CurElem, CurElNm );
```

```
    Elem( xn + i*6 + 2 , yn + j*6     ,  CurElem, CurElNm );
    Elem( xn + i*6     , yn + j*6 + 2 ,  CurElem, CurElNm );
  end;
Rotate( -Pi/2 , true );
for i := -Iter to Iter do
for j := -Iter to Iter do
  begin
    Elem( xn + i*6 + 1 , yn + j*6 + 1 , RotElem, RotElNm );
    Elem( xn + i*6 - 1 , yn + j*6 + 1 , RotElem, RotElNm );
    Elem( xn + i*6 + 1 , yn + j*6 - 1 , RotElem, RotElNm );
    Elem( xn + i*6 + 1 , yn + j*6 + 3 , RotElem, RotElNm );
    Elem( xn + i*6 + 3 , yn + j*6 + 1 , RotElem, RotElNm );
  end;
Rotate( Pi/2 , false);
for i := -Iter to Iter do
for j := -Iter to Iter do
  begin
    Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
    Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
    Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
    Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
    Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
  end;
Rotate( 0 , true );
for i := -Iter to Iter do
for j := -Iter to Iter do
  begin
    Elem( xn + i*6 + 1 , yn + j*6 - 1 ,    RotElem, RotElNm );
    Elem( xn + i*6 - 1 , yn + j*6 - 1 ,    RotElem, RotElNm );
    Elem( xn + i*6 + 1 , yn + j*6 - 3 ,    RotElem, RotElNm );
    Elem( xn + i*6 + 3 , yn + j*6 - 1 ,    RotElem, RotElNm );
    Elem( xn + i*6 + 1 , yn + j*6 + 1 ,    RotElem, RotElNm );
  end;
Rotate( Pi , false);
for i := -Iter to Iter do
for j := -Iter to Iter do
  begin
    Elem( xn + i*6     , yn + j*6     ,   RotElem, RotElNm );
    Elem( xn + i*6 - 2 , yn + j*6     ,   RotElem, RotElNm );
    Elem( xn + i*6     , yn + j*6 - 2 ,   RotElem, RotElNm );
    Elem( xn + i*6 + 2 , yn + j*6     ,   RotElem, RotElNm );
    Elem( xn + i*6     , yn + j*6 + 2 ,   RotElem, RotElNm );
  end;
Rotate( Pi/2 , true );
for i := -Iter to Iter do
for j := -Iter to Iter do
  begin
    Elem( xn + i*6 - 1 , yn + j*6 - 1 , RotElem, RotElNm );
    Elem( xn + i*6 - 3 , yn + j*6 - 1 , RotElem, RotElNm );
    Elem( xn + i*6 - 1 , yn + j*6 - 3 , RotElem, RotElNm );
    Elem( xn + i*6 + 1 , yn + j*6 - 1 , RotElem, RotElNm );
    Elem( xn + i*6 - 1 , yn + j*6 + 1 , RotElem, RotElNm );
  end;
Rotate( -Pi/2 , false);
for i := -Iter to Iter do
for j := -Iter to Iter do
  begin
    Elem( xn + i*6     , yn + j*6     , RotElem, RotElNm );
    Elem( xn + i*6 - 2 , yn + j*6     , RotElem, RotElNm );
    Elem( xn + i*6     , yn + j*6 - 2 , RotElem, RotElNm );
    Elem( xn + i*6 + 2 , yn + j*6     , RotElem, RotElNm );
```

```
          Elem( xn + i*6     , yn + j*6 + 2 , RotElem, RotElNm );
        end;
       Rotate( Pi   , true );
       for i := -Iter to Iter do
       for j := -Iter to Iter do
        begin
         Elem( xn + i*6 - 1 , yn + j*6 + 1 , RotElem, RotElNm );
         Elem( xn + i*6 - 3 , yn + j*6 + 1 , RotElem, RotElNm );
         Elem( xn + i*6 - 1 , yn + j*6 - 1 , RotElem, RotElNm );
         Elem( xn + i*6 + 1 , yn + j*6 + 1 , RotElem, RotElNm );
         Elem( xn + i*6 - 1 , yn + j*6 + 3 , RotElem, RotElNm );
        end;
      end;
13 : begin
     ProgBit := 6 ;
     if  odd(xn) then
      begin
       VirtRot ;
       Elem( xn     , yn     ,  CurElem, CurElNm );
       Elem( xn - 1 , yn - 1 ,  CurElem, CurElNm );
       Elem( xn + 1 , yn - 1 ,  CurElem, CurElNm );
       Elem( xn + 1 , yn + 2 ,  CurElem, CurElNm );
       Elem( xn - 1 , yn + 2 ,  CurElem, CurElNm );
       Elem( xn + 2 , yn     ,  CurElem, CurElNm );
       Elem( xn - 2 , yn     ,  CurElem, CurElNm );
       SRotate( -2*Pi/3 , false);
       Elem( xn     , yn     , RotElem, RotElNm );
       Elem( xn - 1 , yn - 1 , RotElem, RotElNm );
       Elem( xn + 1 , yn - 1 , RotElem, RotElNm );
       Elem( xn + 1 , yn + 2 , RotElem, RotElNm );
       Elem( xn - 1 , yn + 2 , RotElem, RotElNm );
       Elem( xn + 2 , yn     , RotElem, RotElNm );
       Elem( xn - 2 , yn     , RotElem, RotElNm );
       SRotate( 2*Pi/3 , false);
       Elem( xn     , yn     ,  RotElem, RotElNm );
       Elem( xn - 1 , yn - 1 ,  RotElem, RotElNm );
       Elem( xn + 1 , yn - 1 ,  RotElem, RotElNm );
       Elem( xn + 1 , yn + 2 ,  RotElem, RotElNm );
       Elem( xn - 1 , yn + 2 ,  RotElem, RotElNm );
       Elem( xn + 2 , yn     ,  RotElem, RotElNm );
       Elem( xn - 2 , yn     ,  RotElem, RotElNm );
      end
     else
      begin
       VirtRot ;
       Elem( xn     , yn     ,  CurElem, CurElNm );
       Elem( xn - 1 , yn - 2 ,  CurElem, CurElNm );
       Elem( xn + 1 , yn - 2 ,  CurElem, CurElNm );
       Elem( xn + 1 , yn + 1 ,  CurElem, CurElNm );
       Elem( xn - 1 , yn + 1 ,  CurElem, CurElNm );
       Elem( xn + 2 , yn     ,  CurElem, CurElNm );
       Elem( xn - 2 , yn     ,  CurElem, CurElNm );
       SRotate( -2*Pi/3 , false);
       Elem( xn     , yn     , RotElem, RotElNm );
       Elem( xn - 1 , yn - 2 , RotElem, RotElNm );
       Elem( xn + 1 , yn - 2 , RotElem, RotElNm );
       Elem( xn + 1 , yn + 1 , RotElem, RotElNm );
       Elem( xn - 1 , yn + 1 , RotElem, RotElNm );
       Elem( xn + 2 , yn     , RotElem, RotElNm );
       Elem( xn - 2 , yn     , RotElem, RotElNm );
       SRotate( 2*Pi/3 , false);
```

```
        Elem( xn     , yn     ,  RotElem, RotElNm );
        Elem( xn - 1 , yn - 2 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn - 2 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn - 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 2 , yn     ,  RotElem, RotElNm );
        Elem( xn - 2 , yn     ,  RotElem, RotElNm );
      end;
    end;
14 : begin
        ProgBit := 3 ;
        VirtRot ;
        Elem( xn     , yn     ,  CurElem, CurElNm );
        Elem( xn - 1 , yn     ,  CurElem, CurElNm );
        Elem( xn     , yn - 1 ,  CurElem, CurElNm );
        Elem( xn + 1 , yn     ,  CurElem, CurElNm );
        Elem( xn     , yn + 1 ,  CurElem, CurElNm );
          Rotate( Pi   , true );
        Elem( xn + 1 , yn + 1 ,    RotElem, RotElNm );
        Elem( xn     , yn + 1 ,    RotElem, RotElNm );
        Elem( xn + 1 , yn     ,    RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 ,    RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 ,    RotElem, RotElNm );
          SRotate( 2*Pi/3 , false);
        Elem( xn     , yn + 1 ,  RotElem, RotElNm );
        Elem( xn - 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn     , yn     ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn     , yn + 2 ,  RotElem, RotElNm );
          SRotate( -2*Pi/3 , false);
        Elem( xn + 1 , yn + 1 , RotElem, RotElNm );
        Elem( xn     , yn + 1 , RotElem, RotElNm );
        Elem( xn + 1 , yn     , RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 , RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 , RotElem, RotElNm );
          SRotate( Pi/3 , true );
        Elem( xn     , yn     ,   RotElem, RotElNm );
        Elem( xn - 1 , yn     ,   RotElem, RotElNm );
        Elem( xn     , yn - 1 ,   RotElem, RotElNm );
        Elem( xn + 1 , yn     ,   RotElem, RotElNm );
        Elem( xn     , yn + 1 ,   RotElem, RotElNm );
          SRotate( -Pi/3 , true );
        Elem( xn + 1 , yn     ,  RotElem, RotElNm );
        Elem( xn     , yn     ,  RotElem, RotElNm );
        Elem( xn + 1 , yn - 1 ,  RotElem, RotElNm );
        Elem( xn + 2 , yn     ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 1 ,  RotElem, RotElNm );
    end ;
15 : begin
        ProgBit := 2 ;
        VirtRot ;
        Elem( xn     , yn     ,  CurElem, CurElNm );
        Elem( xn - 1 , yn - 2 ,  CurElem, CurElNm );
        Elem( xn - 2 , yn - 1 ,  CurElem, CurElNm );
        Elem( xn - 1 , yn + 1 ,  CurElem, CurElNm );
        Elem( xn + 2 , yn + 1 ,  CurElem, CurElNm );
        Elem( xn + 1 , yn + 2 ,  CurElem, CurElNm );
        Elem( xn + 1 , yn - 1 ,  CurElem, CurElNm );
          Rotate( Pi   , true );
        Elem( xn     , yn     ,  RotElem, RotElNm );
        Elem( xn - 1 , yn - 2 ,  RotElem, RotElNm );
```

```
        Elem( xn - 2 , yn - 1 ,  RotElem, RotElNm );
        Elem( xn - 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn - 1 ,  RotElem, RotElNm );
          SRotate( 2*Pi/3 , false);
        Elem( xn     , yn     ,  RotElem, RotElNm );
        Elem( xn - 1 , yn - 2 ,  RotElem, RotElNm );
        Elem( xn - 2 , yn - 1 ,  RotElem, RotElNm );
        Elem( xn - 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn - 1 ,  RotElem, RotElNm );
          SRotate( -2*Pi/3 , false);
        Elem( xn     , yn     , RotElem, RotElNm );
        Elem( xn - 1 , yn - 2 , RotElem, RotElNm );
        Elem( xn - 2 , yn - 1 , RotElem, RotElNm );
        Elem( xn - 1 , yn + 1 , RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 , RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 , RotElem, RotElNm );
        Elem( xn + 1 , yn - 1 , RotElem, RotElNm );
          SRotate( -Pi/3 , true );
        Elem( xn     , yn     ,  RotElem, RotElNm );
        Elem( xn - 1 , yn - 2 ,  RotElem, RotElNm );
        Elem( xn - 2 , yn - 1 ,  RotElem, RotElNm );
        Elem( xn - 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn - 1 ,  RotElem, RotElNm );
          SRotate( Pi/3 , true );
        Elem( xn     , yn     ,   RotElem, RotElNm );
        Elem( xn - 1 , yn - 2 ,   RotElem, RotElNm );
        Elem( xn - 2 , yn - 1 ,   RotElem, RotElNm );
        Elem( xn - 1 , yn + 1 ,   RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 ,   RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 ,   RotElem, RotElNm );
        Elem( xn + 1 , yn - 1 ,   RotElem, RotElNm );
    end;
16 : begin
        ProgBit := 3 ;
        VirtRot ;
        Elem( xn     , yn     ,  CurElem, CurElNm );
        Elem( xn - 1 , yn     ,  CurElem, CurElNm );
        Elem( xn     , yn - 1 ,  CurElem, CurElNm );
        Elem( xn + 1 , yn     ,  CurElem, CurElNm );
        Elem( xn     , yn + 1 ,  CurElem, CurElNm );
          SRotate( 2*Pi/3 , false);
        Elem( xn     , yn + 1,  RotElem, RotElNm );
        Elem( xn - 1 , yn + 1,  RotElem, RotElNm );
        Elem( xn     , yn     ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 1,  RotElem, RotElNm );
        Elem( xn     , yn + 2,  RotElem, RotElNm );
          SRotate( -2*Pi/3 , false);
        Elem( xn + 1 , yn + 1,  RotElem, RotElNm );
        Elem( xn     , yn + 1,  RotElem, RotElNm );
        Elem( xn + 1 , yn     ,  RotElem, RotElNm );
        Elem( xn + 2 , yn + 1,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 2,  RotElem, RotElNm );
          Rotate( Pi   , false);
        Elem( xn + 1 , yn + 1,  RotElem, RotElNm );
        Elem( xn     , yn + 1,  RotElem, RotElNm );
```

```
        Elem( xn + 1 , yn     ,  RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 ,  RotElem, RotElNm );
          SRotate( -Pi/3 , false);
        Elem( xn + 1 , yn     ,  RotElem, RotElNm );
        Elem( xn     , yn     ,  RotElem, RotElNm );
        Elem( xn + 1 , yn - 1 ,  RotElem, RotElNm );
        Elem( xn + 2 , yn     ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 1 ,  RotElem, RotElNm );
          SRotate( Pi/3 , false);
        Elem( xn     , yn     ,   RotElem, RotElNm );
        Elem( xn - 1 , yn     ,   RotElem, RotElNm );
        Elem( xn     , yn - 1 ,   RotElem, RotElNm );
        Elem( xn + 1 , yn     ,   RotElem, RotElNm );
        Elem( xn     , yn + 1 ,   RotElem, RotElNm );
    end;
17 : begin
        ProgBit := 1 ;
        VirtRot ;
        Elem( xn     , yn     ,  CurElem, CurElNm );
        Elem( xn - 1 , yn - 2 ,  CurElem, CurElNm );
        Elem( xn - 2 , yn - 1 ,  CurElem, CurElNm );
        Elem( xn - 1 , yn + 1 ,  CurElem, CurElNm );
        Elem( xn + 2 , yn + 1 ,  CurElem, CurElNm );
        Elem( xn + 1 , yn + 2 ,  CurElem, CurElNm );
        Elem( xn + 1 , yn - 1 ,  CurElem, CurElNm );
          Rotate( Pi   , true );
        Elem( xn     , yn     , RotElem, RotElNm );
        Elem( xn - 1 , yn - 2 , RotElem, RotElNm );
        Elem( xn - 2 , yn - 1 , RotElem, RotElNm );
        Elem( xn - 1 , yn + 1 , RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 , RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 , RotElem, RotElNm );
        Elem( xn + 1 , yn - 1 , RotElem, RotElNm );
          SRotate( -Pi/3 , false);
        Elem( xn     , yn - 1 ,  RotElem, RotElNm );
        Elem( xn - 2 , yn - 2 ,  RotElem, RotElNm );
        Elem( xn - 1 , yn - 3 ,  RotElem, RotElNm );
        Elem( xn - 1 , yn     ,  RotElem, RotElNm );
        Elem( xn + 2 , yn     ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn - 2 ,  RotElem, RotElNm );
          Rotate( 0  , true );
        Elem( xn     , yn - 1 ,  RotElem, RotElNm );
        Elem( xn - 1 , yn - 3 ,  RotElem, RotElNm );
 Elem( xn - 2 , yn - 2 ,  RotElem, RotElNm );
        Elem( xn - 1 , yn     ,  RotElem, RotElNm );
        Elem( xn + 2 , yn     ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn - 2 ,  RotElem, RotElNm );
          SRotate( 2*Pi/3 , false);
        Elem( xn     , yn     ,  RotElem, RotElNm );
        Elem( xn - 1 , yn - 2 ,  RotElem, RotElNm );
        Elem( xn - 2 , yn - 1 ,  RotElem, RotElNm );
        Elem( xn - 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn - 1 ,  RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 ,  RotElem, RotElNm );
          SRotate( -Pi/3 , true );
        Elem( xn     , yn     ,  RotElem, RotElNm );
        Elem( xn - 1 , yn - 2 ,  RotElem, RotElNm );
```

```
        Elem( xn - 2 , yn - 1 ,  RotElem, RotElNm );
        Elem( xn - 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn - 1 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 ,  RotElem, RotElNm );
          SRotate( Pi/3 , false);
        Elem( xn - 1 , yn     ,   RotElem, RotElNm );
        Elem( xn - 2 , yn - 2 ,   RotElem, RotElNm );
        Elem( xn - 3 , yn - 1 ,   RotElem, RotElNm );
        Elem( xn + 1 , yn + 1 ,   RotElem, RotElNm );
        Elem( xn     , yn - 1 ,   RotElem, RotElNm );
        Elem( xn - 2 , yn + 1 ,   RotElem, RotElNm );
        Elem( xn     , yn + 2 ,   RotElem, RotElNm );
          SRotate( 2*Pi/3 , true );
        Elem( xn - 1 , yn     ,  RotElem, RotElNm );
        Elem( xn - 2 , yn - 2 ,  RotElem, RotElNm );
        Elem( xn - 3 , yn - 1 ,  RotElem, RotElNm );
        Elem( xn - 2 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn + 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn     , yn + 2 ,  RotElem, RotElNm );
        Elem( xn     , yn - 1 ,  RotElem, RotElNm );
          SRotate( -2*Pi/3 , false);
        Elem( xn     , yn     , RotElem, RotElNm );
        Elem( xn - 1 , yn - 2 , RotElem, RotElNm );
        Elem( xn - 2 , yn - 1 , RotElem, RotElNm );
        Elem( xn - 1 , yn + 1 , RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 , RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 , RotElem, RotElNm );
        Elem( xn + 1 , yn - 1 , RotElem, RotElNm );
          SRotate( Pi/3 , true );
        Elem( xn     , yn     ,   RotElem, RotElNm );
        Elem( xn - 1 , yn - 2 ,   RotElem, RotElNm );
        Elem( xn - 2 , yn - 1 ,   RotElem, RotElNm );
        Elem( xn - 1 , yn + 1 ,   RotElem, RotElNm );
        Elem( xn + 1 , yn + 2 ,   RotElem, RotElNm );
        Elem( xn + 2 , yn + 1 ,   RotElem, RotElNm );
        Elem( xn + 1 , yn - 1 ,   RotElem, RotElNm );
          Rotate( Pi   , false);
        Elem( xn + 1 , yn + 1 ,   RotElem, RotElNm );
        Elem( xn     , yn - 1 ,   RotElem, RotElNm );
        Elem( xn - 1 , yn     ,   RotElem, RotElNm );
        Elem( xn     , yn + 2 ,   RotElem, RotElNm );
        Elem( xn + 3 , yn + 2 ,   RotElem, RotElNm );
        Elem( xn + 2 , yn + 3 ,   RotElem, RotElNm );
        Elem( xn + 2 , yn     ,   RotElem, RotElNm );
          SRotate( 4*Pi/3 , true );
        Elem( xn + 1 , yn + 1 ,  RotElem, RotElNm );
        Elem( xn     , yn - 1 ,  RotElem, RotElNm );
        Elem( xn - 1 , yn     ,  RotElem, RotElNm );
        Elem( xn     , yn + 2 ,  RotElem, RotElNm );
        Elem( xn + 3 , yn + 2 ,  RotElem, RotElNm );
        Elem( xn + 2 , yn + 3 ,  RotElem, RotElNm );
        Elem( xn + 2 , yn     ,  RotElem, RotElNm );
    end;
18 : begin     { 01 }
      if StripT = vert  then
       begin
        ProgBit := 15 ;
        VirtRot ;
        for j := -Iter to Iter do
         begin
```

```
            Elem( xn      , yn + j     ,  CurElem, CurElNm );
            Elem( xn      , yn + j - 1 ,  CurElem, CurElNm );
            Elem( xn      , yn + j + 1 ,  CurElem, CurElNm );
          end;
        end
      else
        begin
          ProgBit := 15 ;
          VirtRot ;
          for i := -Iter to Iter do
            begin
              Elem( xn + i     , yn      ,  CurElem, CurElNm );
              Elem( xn + i - 1 , yn      ,  CurElem, CurElNm );
              Elem( xn + i + 1 , yn      ,  CurElem, CurElNm );
            end;
        end;
    end;
19 : begin      { 02 }
      if StripT = vert  then
        begin
          ProgBit := 30 ;
          VirtRot ;
          for j := -Iter to Iter do
            begin
              Elem( xn     , yn + j*2     ,  CurElem, CurElNm );
              Elem( xn     , yn + j*2 - 2 ,  CurElem, CurElNm );
              Elem( xn     , yn + j*2 + 2 ,  CurElem, CurElNm );
            end;
          Rotate( 0    , true);
          for j := -Iter to Iter do
            begin
              Elem( xn     , yn + j*2 - 1 ,  RotElem, RotElNm );
              Elem( xn     , yn + j*2 - 3 ,  RotElem, RotElNm );
              Elem( xn     , yn + j*2 + 1 ,  RotElem, RotElNm );
            end;
        end
      else
        begin
          ProgBit := 30 ;
          VirtRot ;
          for i := -Iter to Iter do
            begin
              Elem( xn + i*2     , yn      ,  CurElem, CurElNm );
              Elem( xn + i*2 - 2 , yn      ,  CurElem, CurElNm );
              Elem( xn + i*2 + 2 , yn      ,  CurElem, CurElNm );
            end;
          Rotate( Pi   , true);
          for i := -Iter to Iter do
            begin
              Elem( xn + i*2 - 1 , yn      ,  RotElem, RotElNm );
              Elem( xn + i*2 - 3 , yn      ,  RotElem, RotElNm );
              Elem( xn + i*2 + 1 , yn      ,  RotElem, RotElNm );
            end;
        end;
    end;
20 : begin      { 03 }
      if StripT = vert  then
        begin
          ProgBit := 15 ;
          VirtRot ;
          for j := -Iter to Iter do
```

```
          begin
            Elem( xn     , yn + j*2     ,  CurElem, CurElNm );
            Elem( xn     , yn + j*2 - 2 ,  CurElem, CurElNm );
            Elem( xn     , yn + j*2 + 2 ,  CurElem, CurElNm );
          end;
        Rotate( Pi  , true);
        for j := -Iter to Iter do
          begin
            Elem( xn     , yn + j*2     ,  RotElem, RotElNm );
            Elem( xn     , yn + j*2 - 2 ,  RotElem, RotElNm );
            Elem( xn     , yn + j*2 + 2 ,  RotElem, RotElNm );
          end;
      end
    else
      begin
        ProgBit := 15 ;
        VirtRot ;
        for i := -Iter to Iter do
          begin
            Elem( xn + i*2     , yn     ,  CurElem, CurElNm );
            Elem( xn + i*2 - 2 , yn     ,  CurElem, CurElNm );
            Elem( xn + i*2 + 2 , yn     ,  CurElem, CurElNm );
          end;
        Rotate( 0   , true);
        for i := -Iter to Iter do
          begin
            Elem( xn + i*2     , yn     ,  RotElem, RotElNm );
            Elem( xn + i*2 - 2 , yn     ,  RotElem, RotElNm );
            Elem( xn + i*2 + 2 , yn     ,  RotElem, RotElNm );
          end;
      end;
  end;
21 : begin        { 04 }
    if StripT = vert  then
      begin
        ProgBit := 30 ;
        VirtRot ;
        for j := -Iter to Iter do
          begin
            Elem( xn     , yn + j*2     ,  CurElem, CurElNm );
            Elem( xn     , yn + j*2 - 2 ,  CurElem, CurElNm );
            Elem( xn     , yn + j*2 + 2 ,  CurElem, CurElNm );
          end;
        Rotate( Pi   , false);
        for j := -Iter to Iter do
          begin
            Elem( xn     , yn + j*2     ,  RotElem, RotElNm );
            Elem( xn     , yn + j*2 - 2 ,  RotElem, RotElNm );
            Elem( xn     , yn + j*2 + 2 ,  RotElem, RotElNm );
          end;
      end
    else
      begin
        ProgBit := 30 ;
        VirtRot ;
        for i := -Iter to Iter do
          begin
            Elem( xn + i*2     , yn     ,  CurElem, CurElNm );
            Elem( xn + i*2 - 2 , yn     ,  CurElem, CurElNm );
            Elem( xn + i*2 + 2 , yn     ,  CurElem, CurElNm );
          end;
```

```
          Rotate( Pi  , false);
          for i := -Iter to Iter do
            begin
              Elem( xn + i*2     , yn     ,  RotElem, RotElNm );
              Elem( xn + i*2 - 2 , yn     ,  RotElem, RotElNm );
              Elem( xn + i*2 + 2 , yn     ,  RotElem, RotElNm );
            end;
        end;
      end;
22 : begin
       if StripT = vert  then
        begin
          ProgBit := 30 ;
          VirtRot ;
          for j := -Iter to Iter do
            begin
              Elem( xn     , yn + j*4     ,  CurElem, CurElNm );
              Elem( xn     , yn + j*4 - 4 ,  CurElem, CurElNm );
              Elem( xn     , yn + j*4 + 4 ,  CurElem, CurElNm );
            end;
          Rotate( Pi   , false);
          for j := -Iter to Iter do
            begin
              Elem( xn     , yn + j*4 + 2 ,  RotElem, RotElNm );
              Elem( xn     , yn + j*4 - 2 ,  RotElem, RotElNm );
              Elem( xn     , yn + j*4 + 6 ,  RotElem, RotElNm );
            end;
          Rotate( Pi   , true );
          for j := -Iter to Iter do
            begin
              Elem( xn     , yn + j*4     ,  RotElem, RotElNm );
              Elem( xn     , yn + j*4 - 4 ,  RotElem, RotElNm );
              Elem( xn     , yn + j*4 + 4 ,  RotElem, RotElNm );
            end;
          Rotate( 0    , true );
          for j := -Iter to Iter do
            begin
              Elem( xn     , yn + j*4 - 2 ,  RotElem, RotElNm );
              Elem( xn     , yn + j*4 - 6 ,  RotElem, RotElNm );
              Elem( xn     , yn + j*4 + 2 ,  RotElem, RotElNm );
            end;
        end
       else
        begin
          ProgBit := 30 ;
          VirtRot ;
          for i := -Iter to Iter do
            begin
              Elem( xn + i*4     , yn     ,  CurElem, CurElNm );
              Elem( xn + i*4 - 4 , yn     ,  CurElem, CurElNm );
              Elem( xn + i*4 + 4 , yn     ,  CurElem, CurElNm );
            end;
          Rotate( 0   , true );
          for i := -Iter to Iter do
            begin
              Elem( xn + i*4     , yn     ,  RotElem, RotElNm );
              Elem( xn + i*4 - 4 , yn     ,  RotElem, RotElNm );
              Elem( xn + i*4 + 4 , yn     ,  RotElem, RotElNm );
            end;
          Rotate( Pi  , false);
          for i := -Iter to Iter do
```

```
                begin
                  Elem( xn + i*4 + 2 , yn     ,  RotElem, RotElNm );
                  Elem( xn + i*4 - 2 , yn     ,  RotElem, RotElNm );
                  Elem( xn + i*4 + 6 , yn     ,  RotElem, RotElNm );
                end;
              Rotate( Pi  , true );
              for i := -Iter to Iter do
                begin
                  Elem( xn + i*4 - 2 , yn     ,  RotElem, RotElNm );
                  Elem( xn + i*4 - 6 , yn     ,  RotElem, RotElNm );
                  Elem( xn + i*4 + 2 , yn     ,  RotElem, RotElNm );
                end;
            end;
          end;
        23 : begin
             if StripT = vert  then
              begin
               ProgBit := 30 ;
               VirtRot ;
               for j := -Iter to Iter do
                begin
                  Elem( xn     , yn + j     ,  CurElem, CurElNm );
                  Elem( xn     , yn + j - 1 ,  CurElem, CurElNm );
                  Elem( xn     , yn + j + 1 ,  CurElem, CurElNm );
                end;
               Rotate( 0    , true );
               for j := -Iter to Iter do
                begin
                  Elem( xn     , yn + j     ,  RotElem, RotElNm );
                  Elem( xn     , yn + j - 1 ,  RotElem, RotElNm );
                  Elem( xn     , yn + j + 1 ,  RotElem, RotElNm );
                end;
              end
             else
          end;
    end;
end.
```

Ж.2. Модуль роботи з формулою мінімального рисунку

```
unit Ornam3z;
interface
uses  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
      ExtCtrls, Buttons, StdCtrls, Menus, ExtDlgs, ComCtrls, Gauges, ToolWin;
  procedure Compile ;
  procedure UpSep( Frml : String ; var AStr : String );
  procedure VarCheck( AStr : String ) ;
  function CheckBrace( AStr : String ):Boolean ;
const
  VarSet = [ 'A' .. 'Z' ] ;
  TpSet  = [ 'a', 'b', 'c', 'd', '@' ] ;
  SgnSet = [ '+', '-', '*', '.' ] ;
type
  TBlockChip = Record
```

```
              b, e : Integer ;
              neg : Boolean ;
             End;
  TBlock = Array[1..3200] of TBlockChip ;
  TVarBlock = Record
              srs : TBlock ;
              len : Integer ;
            End;
var
  VarArr : Array[65..90] of ^TVarBlock ;
  SinErr : Boolean ;
  PrevStr : String ;
implementation
uses Ornam3G, Ornam3R ;
function CheckBrace( AStr : String ):Boolean ;
var i, sr, sl : Integer ;
begin
  sr := 0 ;
  sl := 0 ;
  result := false ;
  if AStr = '' then
    begin
      ShowMessage( ' порожня формула ') ;
      Exit ;
    end;
  for i := 0 to length( AStr ) do
    if AStr[i] = '(' then
      sl := sl + 1
    else if AStr[i] = ')' then
      sr := sr + 1 ;
  if sr < sl then
    ShowMessage( ' непарна ліва дужка ')
  else
  if sr > sl then
    ShowMessage( ' непарна права дужка ')
  else
    result := true ;
end;
procedure UpSep( Frml : String ; var AStr : String );
var
  SepSet : Set of Char ;
  i : Integer ;
begin
  SepSet := [ ' ', #$D, #$A ] ;
  for i := 1 to length( Frml ) do
    if not( Frml[i] in SepSet ) then
      AStr := AStr + Frml[i] ;
end;
function StrToLink( InStr : String ; var flr : Boolean ): TChainEl ;
var
  i, L : Integer ;
  tStr : String ;
begin
  flr := false ;
  L := length( InStr ) ;
{ type ********************************************************************* }
  if InStr[1] = '@' then
    result.tp := fl
  else
  if InStr[1] = '~' then
    if InStr[2] = 'n' then
```

```
      begin
        case InStr[3] of
          'a' : result.tp := _na ;
          'b' : result.tp := _nb ;
          'c' : result.tp := _nc ;
          'd' : result.tp := _nd ;
        end;
        if not ( InStr[3] in [ 'a' .. 'd' ] ) then
          flr := true ;
      end
    else
      begin
        case InStr[2] of
          'a' : result.tp := _ta ;
          'b' : result.tp := _tb ;
          'c' : result.tp := _tc ;
          'd' : result.tp := _td ;
        end;
        if not ( InStr[2] in [ 'a' .. 'd' ] ) then
          flr := true ;
      end
  else
    if InStr[1] = 'n' then
      begin
        case InStr[2] of
          'a' : result.tp := na ;
          'b' : result.tp := nb ;
          'c' : result.tp := nc ;
          'd' : result.tp := nd ;
        end;
        if not ( InStr[2] in [ 'a' .. 'd' ] ) then
          flr := true ;
      end
    else
      begin
        case InStr[1] of
          'a' : result.tp := ta ;
          'b' : result.tp := tb ;
          'c' : result.tp := tc ;
          'd' : result.tp := td ;
        end;
        begin
      Result[i] := Block1[i] ;
      i := i + 1 ;
    end;
  i := i - 1 ;
  j := 1 ;
  while Block2[j].b <> 0 do
    j := j + 1 ;
  len := j - 1 ;
  for j := 1 to len do
    begin
      Result[i+j].b := Block2[len-j+1].e ;
      Result[i+j].e := Block2[len-j+1].b ;
    end;
  ChainQnt := ChainQnt + 1 ;
  Result[i+len].e := Result[i+len].e + 1 ;
  FArr[ChainQnt].tp := pt ;
  FArr[ChainQnt].ln := FArr[ Block1[i].e ].xn - FArr[ Block2[len].e ].xn ;
  FArr[ChainQnt].cl := FArr[ Block1[i].e ].yn - FArr[ Block2[len].e ].yn ;
end;
```

```
function InvertLink( Link : TChainEl ): TChainEl ;
var i, len : Longint ;
begin
  Result := Link ;
  case Result.tp of
    pt : begin
           Result.ln := - Result.ln ;
           Result.cl := - Result.cl ;
         end;
na : Result.tp := _na ;
    nb : Result.tp := _nb ;
    nc : Result.tp := _nc ;
    nd : Result.tp := _nd ;
_na : Result.tp :=  na ;
   _nb : Result.tp :=  nb ;
   _nc : Result.tp :=  nc ;
   _nd : Result.tp :=  nd ;
  end; { of case }
end;
function InvertBlock( Block : TBlock ): TBlock ;
var i, len : Longint ;
begin
  i := 1 ;
  repeat
    Result[i].b := 0 ;
    Result[i].neg := false ;
    i := i + 1 ;
  until Result[i].b = 0 ;
  i := 1 ;
  while Block[i].b <> 0 do
    i := i + 1 ;
  len := i - 1 ;
  for i := 1 to len do
    begin
      Result[len-i+1].neg := not Result[len-i+1].neg ;
      Result[len-i+1].e := Block[i].b ;
      Result[len-i+1].b := Block[i].e ;
    end;
end;
function ReturnLink( CurLink : Integer ):TChainEl ;
begin
  ImageForm.ReFill ;
  Result.tp := pt ;
  Result.ln := FArr[CurLink-1].xn - FArr[CurLink].xn ;
  Result.cl := FArr[CurLink-1].yn - FArr[CurLink].yn ;
end;
function BuildChain( Block : TBlock; Chain : TChain; FromLink : Integer ): TChain;
var
  i, j, k : Integer ;
  flr : Boolean ;
begin
  Result := FArr ;
  i := 1 ;
  k := FromLink + 1 ;
  repeat
    if not Block[i].neg then
      for j := Block[i].b to Block[i].e do
        begin
          Result[k] := FArr[j] ;
          k := k + 1 ;
        end
```

```
    else
      for j := Block[i].b downto Block[i].e do
        begin
          Result[k] := InvertLink( FArr[j] );
          k := k + 1 ;
        end;
    i := i + 1 ;
  until Block[i].b = 0 ;
end;
function StrToBlock( AStr : String; var ChipQnt : Integer ): TBlock ;
var
  TempStr : String ;
  CurChip, i, j : Longint ;
  LastSgn, PrevSgn : Char ;
  NS, flr : Boolean ;
  brc, BegLink, BlockLen : Integer ;
begin
  BegLink := ChainQnt ;
  BlockLen := 0 ;
  i := 1;
  repeat
    Result[i].b := 0 ;
    Result[i].neg := false ;
    i := i + 1 ;
  until Result[i].b = 0 ;
  PrevSgn := '+' ;
  LastSgn := '+' ;
  CurChip := 0 ;
  i := 1 ;
  repeat
    TempStr := '' ;
    j := 1 ;
    CurChip := CurChip + 1 ;
    ImageForm.ReFill ;
    if ( AStr[i] in VarSet )or( ( AStr[i] = '~' )and( AStr[i+1] in VarSet ) ) then
      begin
        NS := ( AStr[i] = '~' ) ;
        i := i + 1 ;
        LastSgn := AStr[i] ;
        if NS then
          case PrevSgn of
           '+' : Result := Join( Result, InvertBlock( VarArr[ord(AStr[i])]^.srs ) );
           '-' : Result := Jemn( Result, InvertBlock( VarArr[ord(AStr[i])]^.srs ) );
           '*' : Result := Joke( Result, InvertBlock( VarArr[ord(AStr[i])]^.srs ) );
           '.', #0 : Exit ;
          end { of case }
        else
          case PrevSgn of
           '+' : Result := Join( Result, VarArr[ord(AStr[i-1])]^.srs );
           '-' : Result := Jemn( Result, VarArr[ord(AStr[i-1])]^.srs );
           '*' : Result := Joke( Result, VarArr[ord(AStr[i-1])]^.srs );
           '.', #0 : Exit ;
          end; { of case }
        if NS then
          if VarArr[ord(AStr[i])]^.len > 1 then
            CurChip := CurChip + VarArr[ord(AStr[i])]^.len
        else
          if VarArr[ord(AStr[i-1])]^.len > 1 then
            CurChip := CurChip + VarArr[ord(AStr[i-1])]^.len ;
        if PrevSgn <> '+' then
          CurChip := CurChip + 1 ;
```

```
    i := i + 1 ;
   end
  else
  if ( AStr[i] = '(' )or( ( AStr[i] = '~' )and( AStr[i+1] = '(' ) ) then
   begin                                  { recurse (...) }
    brc := 1 ;
    NS := ( AStr[i] = '~' ) ;
    i := i + 1 ;
    if NS then
     i := i + 1 ;
    if AStr[i] = ')' then
     brc := brc - 1 ;
    if AStr[i] = '(' then
     brc := brc + 1 ;
    repeat
     TempStr := TempStr + AStr[i] ;
     i := i + 1 ;
     if AStr[i] = ')' then
      brc := brc - 1 ;
     if AStr[i] = '(' then
      brc := brc + 1 ;
    until brc = 0 ;
    i := i + 1 ;
    LastSgn := AStr[i] ;
    if NS then
     case PrevSgn of
      '+' : Result := Join( Result, InvertBlock( StrToBlock( TempStr, BlockLen ) ) );
      '-' : Result := Jemn( Result, InvertBlock( StrToBlock( TempStr, BlockLen ) ) );
      '*' : Result := Joke( Result, InvertBlock( StrToBlock( TempStr, BlockLen ) ) );
      '.', #0 : Exit ;
     end { of case }
    else
     case PrevSgn of
      '+' : Result := Join( Result, StrToBlock( TempStr, BlockLen ) );
      '-' : Result := Jemn( Result, StrToBlock( TempStr, BlockLen ) );
      '*' : Result := Joke( Result, StrToBlock( TempStr, BlockLen ) );
      '.', #0 : Exit ;
     end; { of case }
    if BlockLen > 1 then
     CurChip := CurChip + 1 ;
    if PrevSgn <> '+' then
     CurChip := CurChip + 1 ;
    i := i + 1 ;
   end
  else                { next chip }
   begin
    ChainQnt := ChainQnt + 1 ;
    Result[CurChip].b := ChainQnt ;
    Result[CurChip].e := ChainQnt ;
    repeat
     TempStr := TempStr + AStr[i] ;
     i := i + 1 ;
    until ( AStr[i] in SgnSet )or( i > length( AStr ) ) ;
    LastSgn := AStr[i] ;
    i := i + 1 ;
    if TempStr[1] = '~' then
     begin
      delete( TempStr, 1, 1 );
      case PrevSgn of
       '+' :   FArr[ChainQnt] := InvertLink( StrToLink( TempStr, flr ) ) ;
       '-' : begin
```

```
                FArr[ChainQnt] := ReturnLink( ChainQnt-1 );
                ChainQnt := ChainQnt + 1 ;
                Result[CurChip].e := Result[CurChip].e + 1 ;
                FArr[ChainQnt] := InvertLink( StrToLink( TempStr, flr )) ;
              end;
          '*' : begin
                FArr[ChainQnt] := StrToLink( TempStr, flr ) ; { double-invert }
                ChainQnt := ChainQnt + 1 ;
                Result[CurChip].e := Result[CurChip].e + 1 ;
                FArr[ChainQnt] := ReturnLink( ChainQnt-1 );
              end;
          '.', #0 : Exit ;
        end; { of case }
        if flr then
          begin
            ShowMessage( 'помилка синтаксису: '+TempStr );
            SinErr := true ;
            Exit ;
          end;
      end
    else
      begin
        try
          case PrevSgn of
            '+' :   FArr[ChainQnt] := StrToLink( TempStr, flr ) ;
            '-' : begin
                    FArr[ChainQnt] := ReturnLink( ChainQnt-1 );
                    Result[CurChip].e := Result[CurChip].e + 1 ;
                    ChainQnt := ChainQnt + 1 ;
                    FArr[ChainQnt] := StrToLink( TempStr, flr ) ;
                  end;
            '*' : begin
                    FArr[ChainQnt] := InvertLink( StrToLink( TempStr, flr ) ) ;
                    Result[CurChip].e := Result[CurChip].e + 1 ;
                    ChainQnt := ChainQnt + 1 ;
                    FArr[ChainQnt] := ReturnLink( ChainQnt-1 );
                  end;
            '.', #0 : Exit ;
          end; { of case }
          if flr then
            begin
              ShowMessage( 'помилка синтаксису: '+TempStr );
              SinErr := true ;
              Exit ;
            end;
        except
          begin
            ShowMessage( 'помилка синтаксису: '+TempStr );
            SinErr := true ;
            Exit ;
          end;
        end; { of try }
      end;
  end;
  PrevSgn := LastSgn ;
until LastSgn in [ '.', #0 ] ;
FArr := BuildChain( Result, FArr, BegLink );
ChipQnt := CurChip ;
end;
procedure VarCheck( AStr : String );
var
```

```
  i : Integer ;
  VNm :  'A' .. 'Z' ;
  TempStr : String ;
begin
  VNm := 'A' ;
  i := Pos( '/', AStr ) + 1 ;
  if i <> 1 then  { Pos( '/', AStr ) <> 0 }
    repeat
      TempStr := '' ;
      if AStr[i] in VarSet then
        VNm := AStr[i]
      else
        begin
          ShowMessage( 'Невірне ім`я змінної' );
          SinErr := true ;
          Exit ;
        end;
      i := i + 2 ;  { VNm + '=' }
      repeat
        TempStr := TempStr + AStr[i] ;
        i := i + 1 ;
      until ( AStr[i] in VarSet )or( AStr[i] in [ '.', #0 ] );
      new( VarArr[ord(VNm)] );
      VarArr[ord(VNm)]^.srs := StrToBlock( TempStr, VarArr[ord(VNm)]^.len );
    until AStr[i] in [ '.', #0 ] ;
end;
procedure Compile ;
var
  i : Integer ;
  AStr : String ;
  FBlock : TBlock ;
begin
  SinErr := false ;
  FArr[0].xn := 0 ;
  FArr[0].yn := 0 ;
  i := 1 ;
  repeat
    FArr[i].tp := nl ;
    i := i + 1 ;
  until FArr[i].tp = nl ;
  ChainQnt := 0 ;
  UpSep( FEForm.FEdit.Text, AStr );
  if not CheckBrace( AStr ) then Exit ;
  VarCheck( AStr );
  FBlock := StrToBlock( AStr, i );
  i := 0 ;
  FArr := BuildChain( FBlock, FArr, i );
  if not SinErr then
    begin
      PrevStr := FEForm.FEdit.Text ;
      FEForm.Close ;
    end
  else
    FEForm.FEdit.Text := PrevStr ;
  ImageForm.ReFill ;
end;
end.
```

Додаток З

Інструкція користувача «Редактора орнаментів»

ВСТУП. Комплекс прикладних програм «Редактор орнаментів» призначений для автоматичного формування орнаментів на основі їх математичних формул опису. В його склад входять програми формування мінімальних рисунків та синтезу 17 -ти груп перетворень на площині та 7-ми груп на полосі. За допомогою програмного комплексу можна швидко та якісно розробити довільний орнамент. Разом з тим, використовуючи «Редактор орнаментів», можна ефективно зберігати синтезовані орнаменти в архівованому вигляді (за домогою формул опису, які базуються на структурному методі опису складних зображень).

Програмний пакет застосовується для роботи на комп'ютерах типу IBM PC або сумісних з ними в середовищі Microsoft Windows 3.1 або вище .

Апаратні вимоги до програмно-апаратного комплексу створення орнаментів наступні:

Комп'ютер..... IBM або сумісний

Процесор.....80486

Операційна система.....Windows 3.1 і вище

Ємність оперативної пам'яті..... 16 Мбайт і вище

Ємність дискового простору.....4 Мбайт і вище

Тип дисплею.....SVGA

Прінтер.....графічний

ІНСТАЛЯЦІЙНА ПРОГРАМА. Для інсталяції пакету прикладних програм використовується спеціальна інсталяційна програма INSTALL.EXE. При проведенні операції інсталяції необхідно із середовища Windows вибрати програму інсталяції і дальше слідувати її вказівкам.

ОПИС ІНТЕРФЕЙСУ ПРОГРАМИ. Вікно Автоорнаменту має вигляд представлений на рис. З.1. В верхньому лівому куті вікна знаходиться заголовок.

Нижче заголовка знаходиться рядок меню. Всі команди служать для доступу до різних функцій Автоорнаменту. Щоб вибрати команду із меню потрібно: натиснути ліву кнопку миші на потрібній опції рядка меню і, не відпускаючи її вибрати команду, яку потрібно виконати.

Для полегшення роботи багато операцій, які виконуються з допомогою меню, можна виконати і з допомогою панелі інструментів. Перша - верхня панель інструментів знаходиться під рядком меню, друга - з правої сторони екрану.

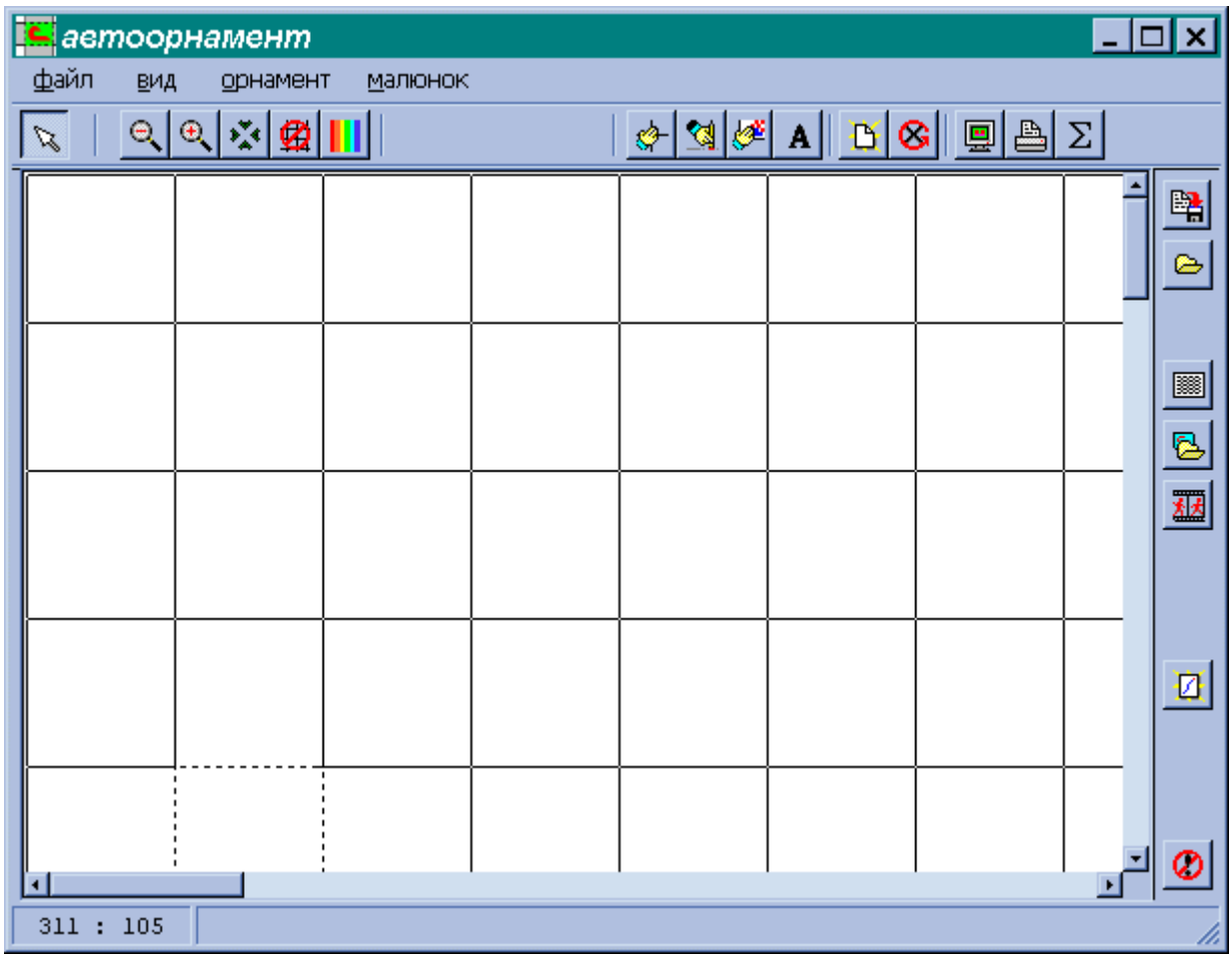


Рис. 3.1. Екран Автоорнаменту.

Опишемо кнопки верхньої панелі інструментів в порядку їх розміщення. Верхня панель показана на рис. 3.2.

Рис. 3.2. Верхня панель інструментів

Кнопки:

Кнопка пустої операції. Натиснена за умовчанням, дозволяє діяти мишею, не побоюючись зробити що-небудь “не так".

Кнопка зменшення масштабу. Коли ця кнопка натиснена кожне натиснення кнопки миші приводить до зменшення масштабу зображення, при цьому точка, на яку вказує миша, переміщається в центр вікна. Цю ж операцію і наступну можна виконати з допомогою команди *масштаб* в рядку меню «Вид». Розкрите меню показане на рис. 3.3.

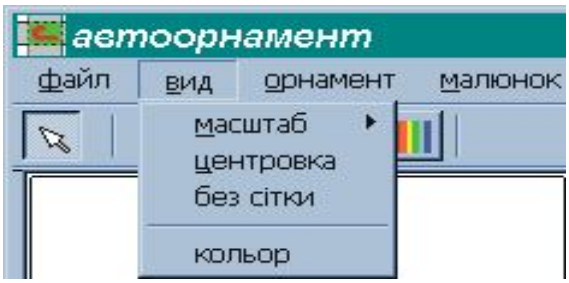


Рис.3.3. Розкрите меню «Вид»

Кнопка збільшення масштабу. Коли ця кнопка натиснена кожне натиснення кнопки миші приводить до збільшення масштабу зображення, при цьому точка, на яку вказує миша, переміщається в центр вікна.

Кнопка « в центр ». Коли ця кнопка натиснена кожне натиснення кнопки миші приводить до переміщення точки, на яку вказує мишу, в центр вікна. Цю операцію можна виконати з допомогою команди *центровка* в рядку меню «Вид». Див. рис. 3.3.

Кнопка « сховати/показати сітку ». Коли ця кнопка натиснена сітка схована. Цю операцію можна виконати з допомогою команди *без сітки* в рядку меню «Вид». Див. рис. 3.3.

Кнопка «колір ». При натисненні відкривається діалогове вікно вибору кольору. Колір фону орнаменту змінюється на вибраний в діалоговому вікні. Цю операцію можна виконати з допомогою команди *колір* в рядку меню «Вид». Див. рис. 3.3.

Кнопка « робота з сіткою ». Щоб змінити розмір сітки натискуємо цю кнопку, підводимо покажчик миші до правого нижнього кута одного з осередків сітки, натискуємо праву кнопку миші, розтягуємо сітку до необхідного розміру і відпускаємо кнопку миші. Щоб змінити розташування сітки аналогічні дії проробляємо тільки з допомогою лівої кнопки миші.

 Кнопка « перемістити ». Якщо потрібно змінити розташування групи натискуємо цю кнопку, потім натискуємо правою кнопкою миші на потрібній групі, поки не буде вибрана саме вона (група вибрана, коли її елементарні малюнки стали чорно-білими); вибравши групу, натискуємо ліву кнопку миші, перетягуємо групу і відпускаємо кнопку миші.

 Кнопка «робота з елементом». Коли ця кнопка натиснена можна поставити нову групу елементів.

 Кнопка « мітка ». Щоб вставити в орнамент текстовий рядок натискуємо цю кнопку, натисненням миші ставимо смугу рядка і вводимо текст. Ввівши рядок натискуємо “Enter". Якщо потрібно змінити шрифт двічі натискуємо мишею на кнопці і вибираємо в діалоговому вікні, що відкрилося параметри шрифту.

 Кнопка « очистити ». Дозволяє очистити створений орнамент. Цю операцію можна виконати з допомогою команди *очистити* в рядку меню «Орнамент». Розкрите меню зображено на рис. 3.4.

 Кнопка « знищити останнє ». Дозволяє знищити поставлений останнім елемент орнаменту.

Кнопка « перегляд ». Дозволяє переглянути весь орнамент в повноекранному режимі.

Рис.3.4. Розкрите меню Орнамент.

Кнопка « друк ». Щоб роздрукувати орнамент натискуємо цю кнопку. Якщо потрібно роздрукувати весь орнамент вибираємо у вікні, що з'явилося опцію « друк всього орнаменту» і натискуємо кнопку « друк ». Якщо потрібно роздрукувати фрагмент вибираємо опцію « друк прямокутного фрагмента»(рис. 3.5), натискуємо кнопку « відмітити фрагмент », в основному вікні натискуємо кнопку миші, розтягуємо відмічений штриховкою прямокутник фрагмента і відпускаємо кнопку миші. Якщо відмічений фрагмент нас влаштовує натискуємо кнопку «друк», в противному разі знову натискуємо «відмітити фрагмент». Якщо немає необхідності роздруковувати натискуємо «відміна друку». Цю операцію можна виконати з допомогою команди *друк* в рядку меню «Малюнок», зображеному на рис. 3.6.

Рис.3.5. Розкрите меню Друк.

Рис. 3.6. Розкрите меню «Малюнок».

Кнопка «формула». Дозволяє переглянути формулу орнаменту. При натискуванні цієї кнопки появляється формула в окремому вікні, яке має вигляд як на рис. 3.7. Цю операцію можна виконати з допомогою команди *формула* в рядку меню «Малюнок». Див. рис.3.6.

Формула орнаменту це послідовність записів (розділених пустими рядками) кожна з яких містить дані про одну орнаментальну групу. Запис включає три рядки (рис. 3.7), середній рядок може розбиватися на окремі рядки якщо він не вміщується в

один).

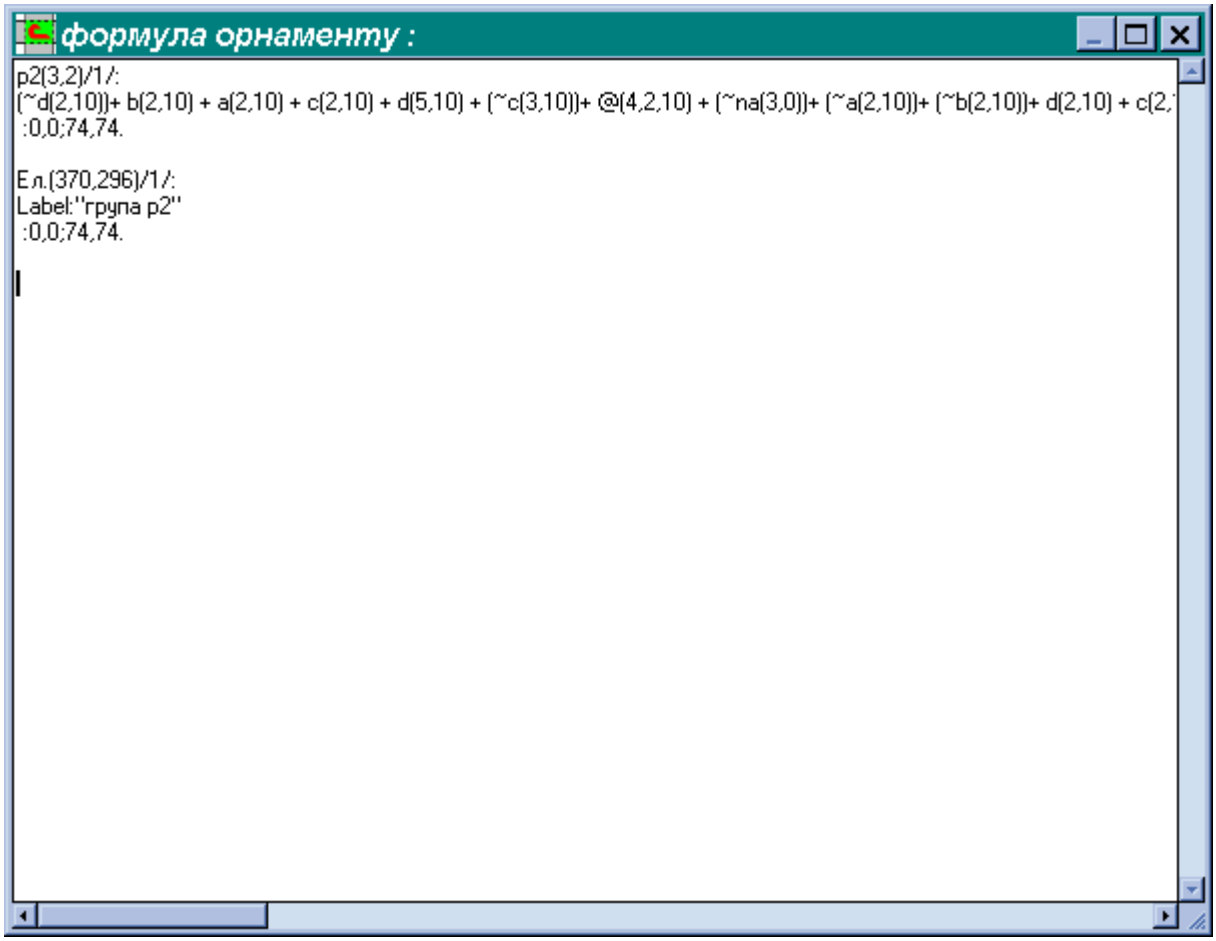


Рис. 3.7. Вікно формули орнаменту.

У першому рядку записані:

1. Позначення групи (наприклад, p1, pmg і т. д.).
2. У дужках номери рядка і стовпця клітки (координати), в якій поставлений перший елемент групи (тобто клітки, в якій знаходився покажчик миші, коли виставлялася група.

Відлік кліток проводиться з лівої верхньої клітинки вздовж стрілок, як показано на рис.3.8.

3. У похилих кількість ітерацій, тобто скільки разів первинна група була розмножена у всі сторони, або кількість розмножених рапортів.

У другому рядку:

1. Якщо в директорії, з якої взятий елементарний малюнок, є файл з тим же ім'ям, але з розширенням “txt", то в рядок (або в декілька рядків) буде записаний вміст txt-файла.

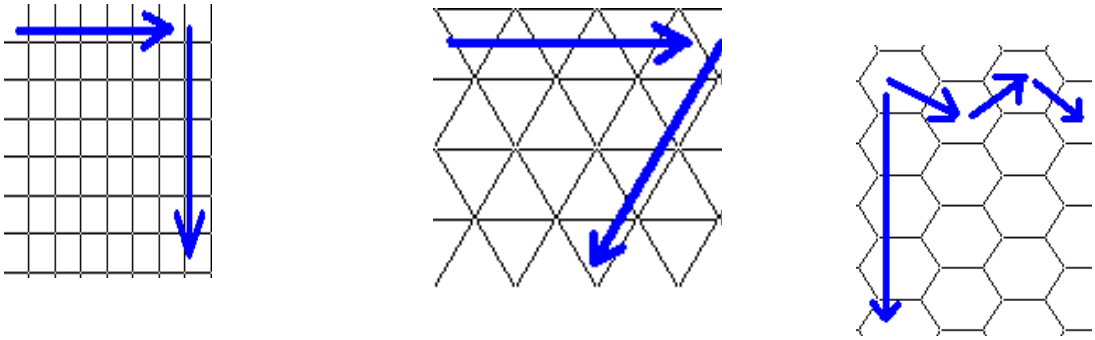


Рис.3.8. Відлік координат першого елемента групи: а) у прямокутній сітці; б) у трикутній сітці; в) у шестикутній сітці.

Якщо в директорії такого файла немає, то буде записане ім'я файла малюнка (і повний шлях до нього).

Якщо дана група це мітка (тобто текстовий рядок), то замість імені буде записано: “Label:" і в лапки текст мітки.

2. Якщо елементарний малюнок був завантажений з масштабуванням по сітці, то після пропуску і косою буде записаний розмір сітки, під яку був промасштабований рисунок.

У третьому рядку:

1. Поточний зсув сітки від первинного положення по вертикалі і по горизонталі.
2. Поточний розмір сітки (після крапки з комою).

Опишемо кнопки правої панелі. Вона зображена на рис. 3.9. (в поверненому на бік вигляді).

Рис. 3.9. Права панель.

Кнопка «зберегти орнамент». Дозволяє зберегти орнамент в файл розширення “orn". Цю операцію можна виконати з допомогою команди «зберегти орнамент» в рядку меню Файл, зображеному на рис. 3.10.

Кнопка « відкрити орнамент ». Дозволяє завантажити орнамент з файла розширення “orn". Цю операцію можна виконати з допомогою команди «відкрити орнамент» в рядку меню Файл (див. рис. 3.10).

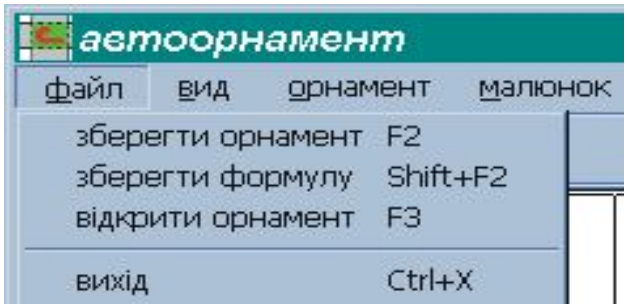


Рис. 3.10. Розкрите меню Файл.

Кнопка « вибір сітки ». Дозволяє встановити тип сітки з трьох даних: прямокутного, трикутного і шестикутного. Цю операцію можна виконати з допомогою команди *сітка* в рядку меню «Орнамент» (див. рис. 3.4).

Кнопка «вибір елементу». Дозволяє завантажити малюнок з bmp-файла (малюнок повинен бути не більшим, ніж 100x100 пікселів). Цю операцію можна виконати з допомогою команди *елемент* в рядку меню «Орнамент» (див. рис. 3.4).

Кнопка «вибір заповнення». Дозволяє вибрати тип заповнення з даних. Цю операцію можна виконати з допомогою команди *заповнення* в рядку меню «Орнамент» (див. рис. 3.4). При натисненні даної кнопки появляється діалогове вікно, зображене на рис. 3.11. Мишею двічі натискаємо на потрібній групі орнаменту. Після цього діалогове вікно зникає, а при натисненні на будь-який осередок сітки (рис. 3.1), при діючій кнопці «робота з елементом», буде формуватися орнамент вибраної групи з мінімальним рисунком, який задано з допомогою попередньої кнопки «вибір елементу».

Кнопка « малюнок ». Дозволяє створити елементарний малюнок. Цю операцію можна виконати з допомогою команди *створити елемент* в рядку меню «Малюнок» (див. рис. 3.6). При натисненні даної кнопки появляється діалогове вікно, зображене на рис. 3.12.

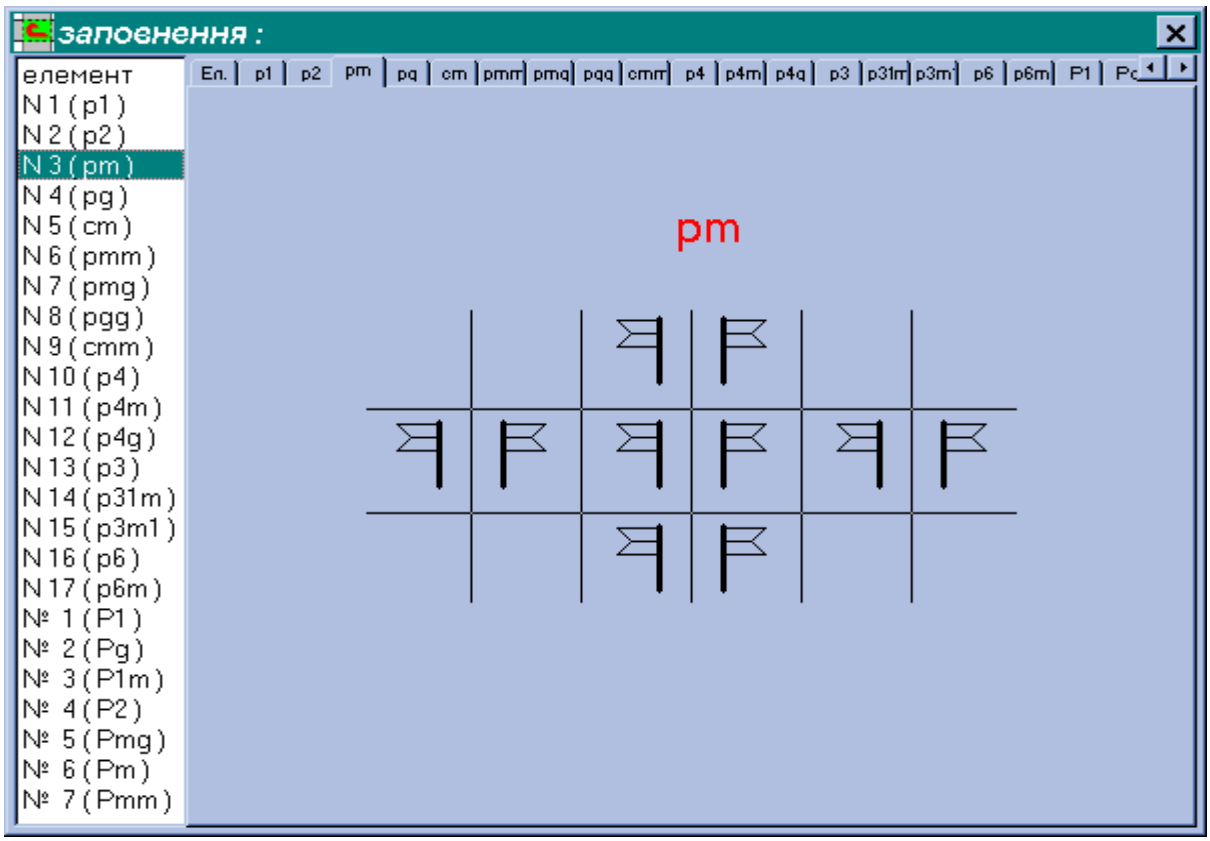


Рис. 3.11. Діалогове вікно «Вибір заповнення».

Кнопка « вихід » - вихід з програми.

Окремо зупинимося на діалоговому вікні «Малюнок». В верхньому лівому куті вікна знаходиться заголовок. Нижче заголовка знаходиться рядок меню. Ще нижче панель інструментів. Основна частина діалогового вікна ділиться на три частини. Зліва розміщена панель кольорів. Щоб вибрати потрібний колір необхідно натиснути лівою кнопкою миші по відповідному кольору. Посередині розміщене поле, в якому будується мінімальний рисунок (алгоритм побудови описано в підрозділі 4.1). Це є сітчаста область, розміри якої визначаються при відкритті діалогового вікна. Справа знаходиться область, в якій висвічується формула побудови мінімального рисунку. Побудова здійснюється в базисі операцій {~, ⊕} (див. підрозділ 3.2).

Примітка. В формулі операцію + слід читати як ⊕ (див. підрозділ 2.2).

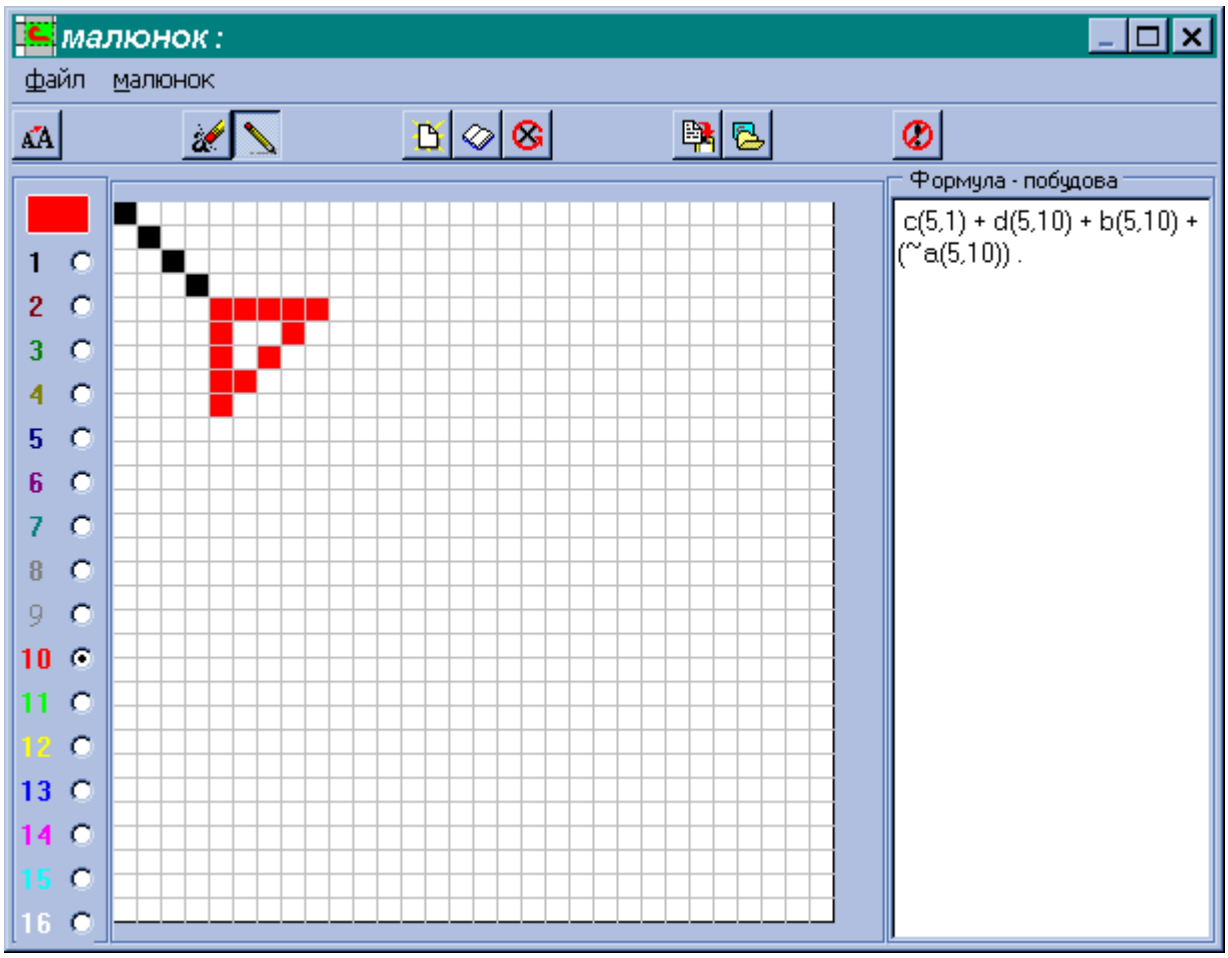


Рис. 3.12. Діалогове вікно «Малюнок».

Опишемо призначення кнопок на панелі інструментів діалогового вікна «Малюнок» (рис. 3.13).

Рис. 3.13. Панель інструментів діалогового вікна «Малюнок».

 Кнопка «редагування формули». Дозволяє змінити поточний елементарний малюнок, редагуючи його формулу.

 Кнопка « заливання ». Коли ця кнопка натиснена можна натисненням миші “залити" поточним кольором область малюнка, обмежену точками інших кольорів.

 Кнопка « лінія ». Коли ця кнопка натиснена можна малювати (стьобанням по вертикалі, по горизонталі і по діагоналі) лінії: при натисненні лівою кнопкою миші

малюється лінія поточного кольору, при натисненні правою кнопкою “прозора лінія".

 Кнопка « новий малюнок » - при натисненні створюється новий малюнок, (дані про старий малюнок не зберігаються !).

 Кнопка « очистити малюнок » - при натисненні створюється новий (“чистий") малюнок того ж розміру, що і попередній.

 Кнопка « назад » - при натисненні забирається остання лінія або заливання.

 Кнопка « зберегти малюнок » - зберегти створений малюнок в bmp-файл; одночасно формула малюнка зберігається в файл з тим же ім'ям, але з розширенням “txt".

 Кнопка « відкрити малюнок » - відкрити для сканування bmp-файл.

 Кнопка « вийти » - вихід з вікна « малюнок ».

Автоорнамент - простий в освоєнні і застосуванні програмний продукт. Працювати в ньому досить легко і приємно.

ЗАТВЕРДЖУЮ
Директор
СП «Комекспо-Київ»

_____Євтушенко О.В.

“____”_____1999 р.

АКТ

про впровадження результатів дисертаційної роботи Березької Катерини Миколаївни “Моделювання та синтез складних зображень симетричної структури ”

Укладений в тім, що запропоновані методи, алгоритми і програмні засоби синтезу складних зображень-орнаментів впроваджені з метою реалізації на базі швейних машин фірми «BROTHER» моделей «PE -100» і «Super Galaxie» орнаментів-вишивок. Запропонований підхід та алгоритми дали можливість принципово по-новому підійти до виготовлення оригінальних зображень в машинній реалізації вишивок на тканині.

Результати дисертаційної роботи Березької К.М. мають вагоме значення і показали високу ефективність при їх застосуванні.

Технічний директор Карамаш О.А.

Дизайнер Журавель А.В.

ЗАТВЕРДЖУЮ
Заступник директора
Державного науково-дослідного
інституту інформаційної інфраструктури

Бунь Р.А.

“____”_____1999 р.

АКТ

про впровадження результатів дисертаційної роботи Березької Катерини Миколаївни “Моделювання та синтез складних зображень симетричної структури ”

Укладений в тім, що результати по моделюванню і синтезу складних зображень використовуються в науково-дослідній тематиці Державного науково-дослідного інституту інформаційної інфраструктури в технічних розробках, пов’язаних з реалізацією проблемно-орієнтованих систем на базі комп’ютерів з метою попередньої обробки зображення: сегментації, виділення окремих частин зображень.

Результати дисертаційної роботи Березької К.М. використовуються в рамках Програми робіт «Розробка інформаційно-аналітичної системи супроводу інтеграції України в міжнародне співтовариство».

Запропонований підхід до синтезу та моделювання складних зображень дає можливість здійснювати ефективно обробку напівтонових зображень.

Заступник директора ДНДІІІ Стех С.М.

Інженер Хавалко В.

ЗАТВЕРДЖУЮ
Проректор з навчальної роботи
Тернопільської академії
народного господарства
_____ проф. Журавель Г.П.
«___» _____ 1999 р.

АКТ
про використання результатів дисертаційної роботи
Березької Катерини Миколаївни
“Моделювання та синтез складних зображень симетричної структури”

Ми, що нижче підписалися, завідувач кафедри спеціалізованих комп’ютерних систем (СКС), завідувач лабораторіями кафедри СКС підтверджуємо, що результати наукових розробок і досліджень Березької К.М. використовуються в навчальному процесі для підготовки спеціалістів за спеціальністю «Комп’ютерні системи та мережі» при проведенні практичних та лабораторних робіт.

Вважаємо, що й надалі доцільно застосовувати їх в навчальному процесі.

Завідувач кафедри спеціалізованих
комп’ютерних систем, д.т.н., доцент Карпінський М.П.

Завідувач лабораторіями
кафедри СКС Содомора Я.В.