УДК 004.04:004.92
Коцюба А.Ю., Сітовський В.О.
Луцький національний технічний університет

РЕАЛІЗАЦІЯ ТЕОРІЇ КЛІТИННИХ АВТОМАТІВ В ПОПІКСЕЛЬНІЙ ОБРОБЦІ РАСТРОВОЇ ГРАФІКИ

Коцюба А.Ю., Сітовський В.О. Реалізація теорії клітиних автоматів в попіксельній обробці растрової графіки. В статті розглянуто теорію абстрактних автоматів (зокрема, теорію клітинних автоматів) та описано основні аспекти її реалізації в програмуванні на прикладі технології попіксельної обробки графіки, що базується на понятті клітинного автомата. Обгрунтовано доцільність використання та впровадження розроблених продуктів на основі теорії клітинних автоматів в середовищі CodeGearRADStudioC++ Builder 2009 для вирішення поставлених цілей та завдань.
Ключові слова: C++ Builder, попіксельна обробка графіки, клітинний автомат.

Коцюба А.Ю., Ситовский В.А. Реализация теории клеточных автоматов в попиксельной обработке растровой графики. В статье рассмотрена теория абстрактных автоматов (в частности, теория клеточных автоматов) и описаны основные аспекты её реализации в программировании на примере технологии попиксельной обработки графики, основанной на понятии клеточного автомата. Обоснована целесообразность использования и внедрения разработанных продуктов на основе теории клеточных автоматов в среде CodeGearRADStudioC ++ Builder 2009 для решения поставленных целей и задач.
Ключевые слова: C++ Builder, попиксельная обработка графики, клеточный автомат.

Kotsiuba A.Yu, Sitovskyi V.O. Realization of the theory of cellular automata for pixel graphics processing. The theory of abstract automata (in particular, the theory of cellular automata) is considered in the article. It describes the main aspects of its implementation in programming using the example of pixel graphics processing technology, based on the concept of a cellular automaton. The expediency of using and implementing the developed products based on the theory of cellular automata in the CodeGearRADStudioC ++ Builder 2009 environment is substantiated for solving goals and objectives set.
Keywords: C++ Builder, pixel graphics processing, cellular automaton.

Вступ. На часі однією з найбільш перспективних є технологія подійно-орієнтованого програмування, яка базується на використанні математичного апарату теорії абстрактних автоматів. Особливо значних результатів можна досягнути, поєднуючи цю технологію з іншими відомими – наприклад, з технологією об'єктно-орієнтованого програмування. Це поєднання технологій має широкий спектр застосування в різних галузях знань: від астрономічних фотографій, медицини та робототехніки, і закінчуючи контролем якості в промисловості. Слід зазначити, що теорія абстрактних автоматів передбачає великий різновид класів автоматів і найбільш використовуваними в даному напрямку серед них є клітинні автомати. Однією із найвідоміших класичних реалізацій даного математичного апарату в програмуванні є гра "Життя". Вона реалізована за принципом ітераційної зміни кольору в клітинках (з чорного на білий або навпаки) в залежності від кольорів в сусідніх клітинках. Виникає ідея розвитку даного алгоритму в напрямках: замість клітинки будемо брати окремий піксель растрового зображення; замість двох кольорів використаємо палітру RGB, у якій налічується 256^3 кольорів. Якщо реалізувати цю ідею в напрямку обробки цифрових зображень, то з'явиться можливість вирішувати широкий клас практично значимих задач пов'язаних з теорією обробки цифрових зображень.

Проблема яку слід розв'язати полягає у тому що, розроблений в середовищі *C++ Builder* програмний продукт обробки растрових зображень, який базується на понятті клітинного автомата повинен вміти зчитувати вхідну інформацію двома способами: за допомогою функціональної залежності, яка формує растрове зображення та за допомогою відкриття уже готового зображення в різних форматах. Далі в залежності від кольору в сусідніх пікселях повинні змінюватись кольори у всіх пікселях вхідного зображення. Цей процес можна ітеративно повторювати будь-яку кількість разів. При цьому слід реалізувати автоматичне збереження всіх результуючих зображень у вибраному користувачем форматі.

Для вирішення поставлених завдань та реалізації вищеописаного підходу в алгоритмах обробки растрових зображень розробимо в середовищі візуального програмування *CodeGearRAD-Studio C++ Builder 2009* програмний продукт і забезпечимо у ньому виконання наступних функцій:

- завантаження растрового зображення;
- перетворення зображення у двоколірне;
- обробки зображення за допомогою алгоритму усереднення з вагою;
- обробки зображення за допомогою алгоритму, який порівнює кожен піксель з сусідами, і в разі подібності замінює його на подібний;

- обробки зображення за допомогою алгоритму виключення хибних пікселів;
- обробки двоколірного зображення за алгоритмом, запропонованим у класичній грі "Життя";
- автоматичне чи ручне збереження результату обробки на будь-якому етапі роботи.

Для виконання вищевказаних функцій служать основні алгоритми:

- алгоритм, що допомагає користувачу вибирати різні два кольори так, щоб один з них був темним, а другий світлим;
- алгоритм попіксельного зчитування зображення;
- алгоритм збереження користувачем результуючого зображення у будь-якому з допустимих діалогом збереження форматі;
- алгоритм формування зображення за допомогою функції від координат пікселя $f(i,j)\rightarrow \rightarrow TColor$;
- вищеописані алгоритми обробки зображення.

Аналіз досліджень в напрямку обробки растрового зображення показує, що на сьогоднішній день таких алгоритмів та середовищ їхньої реалізації існує величезна кількість. Розроблений в даній роботі проект не призначений для конкуренції з ними, а його **метою** є лише отримання навиків та знань в напрямку створення повноцінних візуальних програмних додатків, за допомогою яких в перспективі можна було б, наприклад, наочно та зрозуміло для студентів проводити апробацію та модернізацію відомих в літературі методів обробки графіки.

Термін "*клітинні автомати*" (КА) почав використовуватись у середині XX ст. для позначення сукупності залежних елементів із заданими станами і правилами, за допомогою яких стани цих елементів і залежності між ними змінюються в часі. Час і стани при цьому дискретні. Використання описаних моделей для формального моделювання самовідтворюваних організмів вперше запропоновано в роботі Фон Неймана. Елементи клітинних автоматів запропоновано представити одновимірними або двовимірними нескінченними прямокутними таблицями. Стан елемента змінюється в залежності від його стану і від стану найближчих сусідів.

Класичні клітинні автомати в загальному випадку відповідають наступним критеріям:

- зміна значень всіх клітинок відбуваються одночасно після обчислення нового стану кожної клітинки решітки. Інакше порядок перебору клітин решітки при проходження ітеративного процесу суттєво впливав би на результат;
- решітка однорідна;
- взаємодії локальні. Лише околишні клітинки (як правило, сусідні) здатні вплинути на дану клітинку;
- множина станів клітинки скінченна. Ця умова потрібна, щоб для отримання нового значення стану клітини треба було виконати скінченну кількість операцій (але це не заважає використовувати клітини для зберігання чисел із плаваючою комою для розв'язку прикладних задач).

Розроблений програмний продукт складається з: *3*-х модулів (форм: головної, функціонального та ітераційного налаштування). Інтерфейс користувача досить простий та інтуїтивно зрозумілий. Для скорочення викладок про процес розробки та етапи користування опишемо основні можливості програми за допомогою чотирьох прикладів. Першим прикладом побудуємо двоколірне зображення на основі "реальної" (тут мається на увазі – фото власного виробництва) фотографії розмірністю 1760x1060, зображеної на рис. 1.

Рисунок 1 – Вигляд вхідної фотографії розмірністю 1760x1060

Приклад 1. Спочатку відкриємо дане фото за допомогою розробленої нами програми. Далі натискаємо кнопку, на якій зображено чорно-білі нігті, і вибираємо два кольори, як це описано на рис. 2.

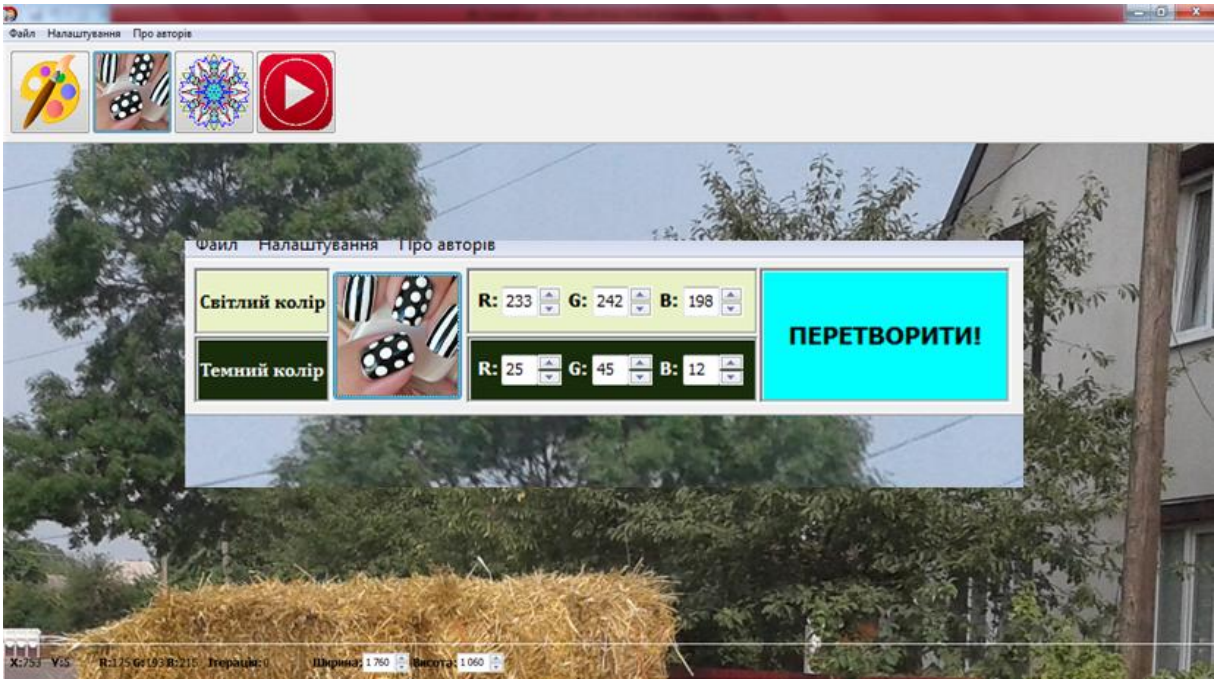


Рисунок 2 – Налаштування двох кольорів для перетворення зображення у двоколірне

Зауважимо, що при виборі кольорів програма відслідковує, щоб темний колір не став світлим, а світлим – відповідно темним. Результат обробки збережемо та подамо на рис. 3.

Рисунок 3 – Двоколірний результат обробки фотографії розмірністю 1760x1060

Очевидно, що таких результатів неважко досягнути за допомогою відомих растрових редакторів, але в даному програмному проекті нам вигідно було реалізувати власну вищеописану обробку, щоб далі на отриманому результаті перевіряти різні алгоритми, які розраховані в першу чергу на два кольори (найвідоміший серед таких алгоритмів є класичний алгоритм гри "Життя"). Оскільки обробка такої розмірності займає близько 10 хв. (а даний алгоритм перетворення досить примітивний), то надалі обмежимося значно меншою розмірністю.

В наступному прикладі запропоновано розглянути дві функціональні заливки.

Приклад 2. Покажемо як зробити заливку фону за допомогою функції $f(i,j) \rightarrow TColor$ (Налаштування-Функціональне налаштування). Розглянемо дві заливки: випадкову і таку, що задається функцією, яку подано на рис. 4.

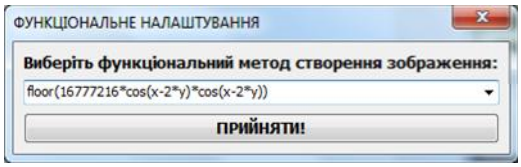


Рисунок 4 – Вигляд діалогового вікна "Функціональне налаштування"

За умовчанням застосовується генератор псевдовипадкових чисел (ГПВЧ), тобто для кожного пікселя з множини потужністю 2^{24} всіх можливих кольорів програма сама вибирає потрібний. Очевидно, що результат є нестабільним і в нашому випадку матиме вигляд як на рис. 5 ліворуч. А на рис. 5 праворуч покажемо результат заливки, що задається функціональним налаштуванням на рис. 4.

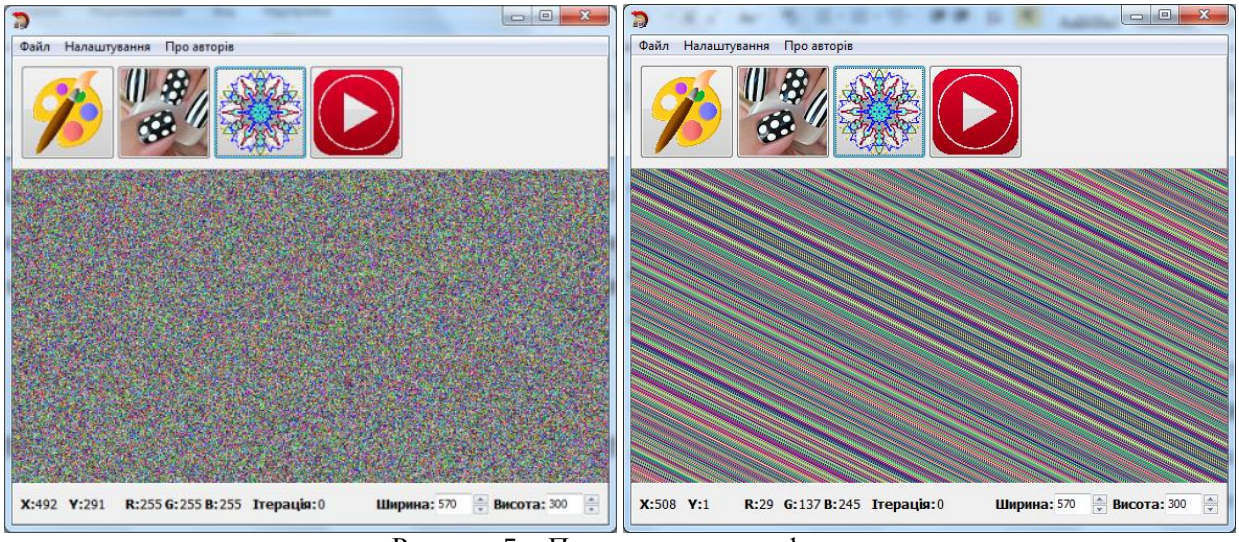


Рисунок 5 – Приклади заливок фону

Очевидно, що пошук різного роду заливок, зводиться до експериментів, які обмежуються лише математичною фантазією.

В наступних прикладах покажемо результати деяких алгоритмів обробки зображень.

Приклад 3. Спочатку розкажемо, як користуватися алгоритмом усереднення з вагою. Для цього приймемо, що в ітераційному налаштуванні були встановленні значення параметрів, заданих в ітераційному налаштуванні (рис. 6).

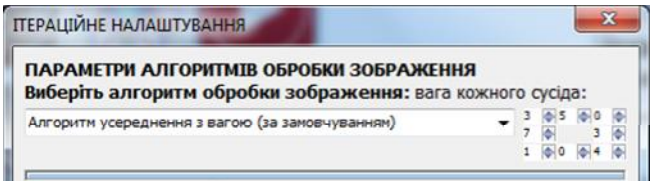
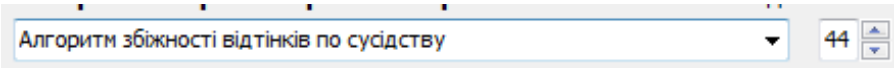


Рисунок 6 – Вигляд діалогового вікна "Ітераційне налаштування"

По-суті все зводиться до підрахунку середнього арифметичного з вибірки, у якій кожен елемент може зустрічатися декілька разів. Зауважимо, що випадок, коли кожна вага є нульовою, не зашкодить роботі алгоритму, просто у цьому випадку програма навіть не почне працювати і видасть повідомлення про помилку.

На рис. 7 показано результат після декількох ітерацій обробки зображення за допомогою алгоритму усереднення з вагою при заданих нами значеннях параметрів для випадково вибраної фотографії.

Далі опишемо роботу 

і покажемо, як він працює при вказаному нами максимальному відхиленні "44". І аналогічно, як і у попередньому випадку, проведемо декілька ітерацій та отримаємо результат (див. рис. 8).

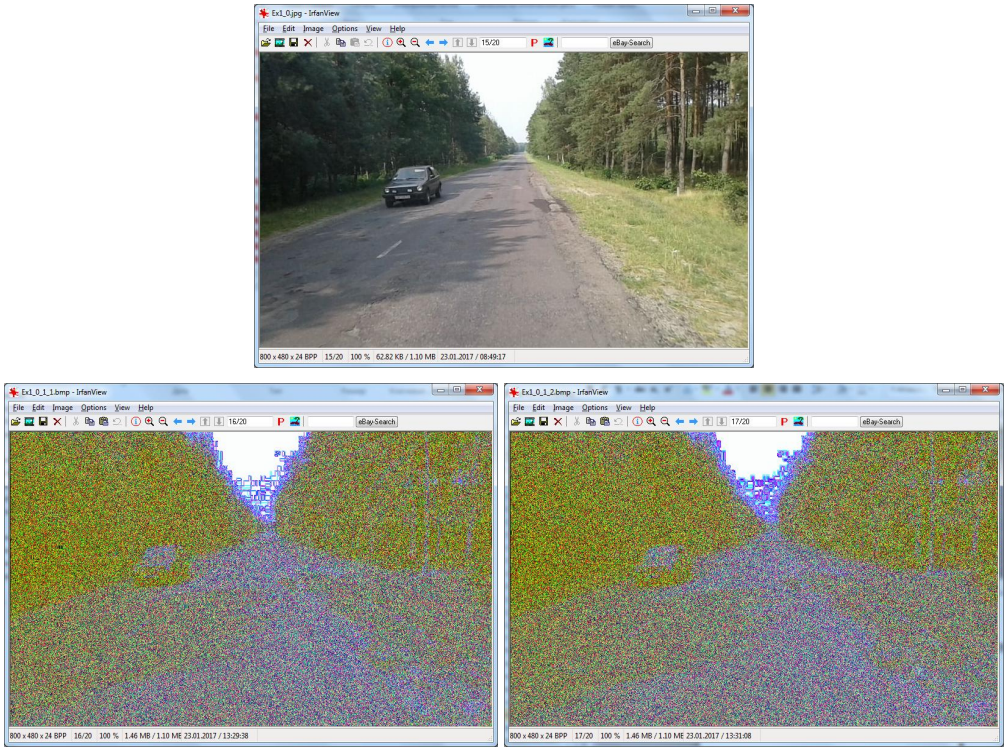


Рисунок 7 – Результати обробки перших двох ітерацій алгоритму усереднення з вагою

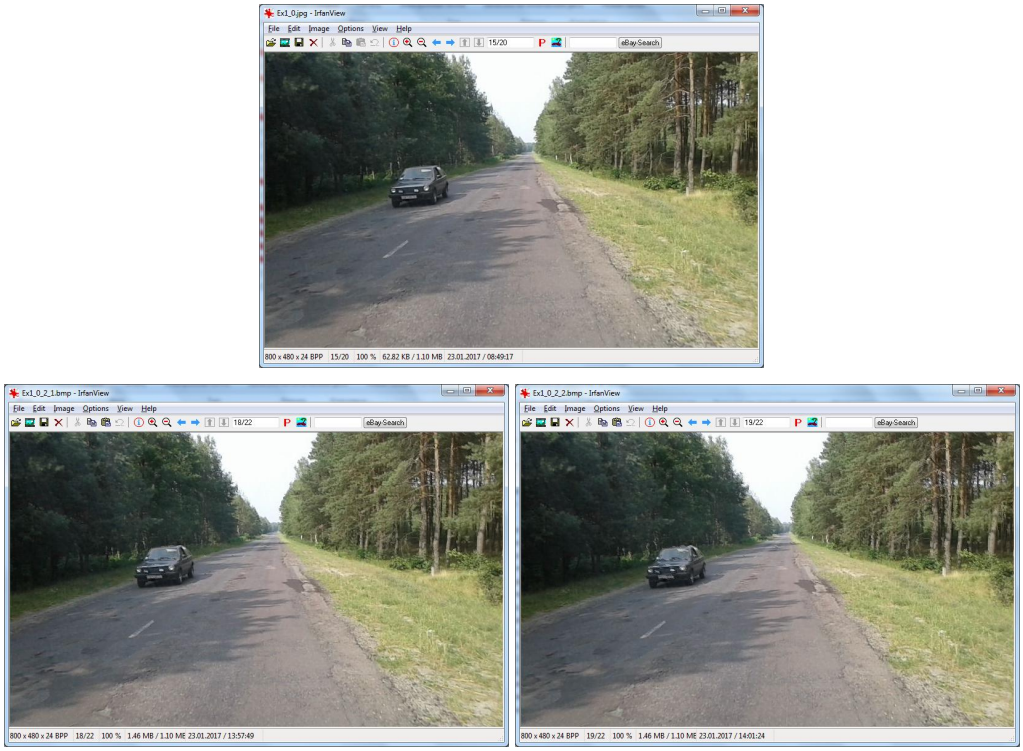


Рисунок 8 – Результати обробки перших двох ітерацій алгоритму збіжності відтінків

Очевидно, що другий алгоритм обробки повільніше спотворює зображення, ніж перший. Виникає питання: для чого потрібно спотворювати фотографію. На перший погляд питання доречне. Але як створювати обернені алгоритми, якщо не буде можливість перевірки. Складання оберненого алгоритму це складна задача, яка зводиться до розв'язування системи великої кількості здебільшого нелінійних (але не для випадку усереднення) рівнянь або нерівностей з великою кількістю невідомих. З точки зору сучасних технологій – це складна проблема, але не на стільки, щоб її не можна було би вирішити.

Приклад 4. В останньому прикладі зупинимося на алгоритмі, який буде знищувати спотворення зображення у вигляді хибних пікселів. Назвемо його алгоритмом знищення хибних пікселів, що мають групи з подібних сусідів. По суті цей алгоритм шукає такі пікселі, з обох сторін, якого є скупчення близьких за кольором сусідів. При цьому колір цього пікселя повинен відрізнятися від кольорів груп сусідів (як правило це дві групи по 3 або 4 пікселя, розташованих поруч), а кожна з цих груп може бути "ворожою" не лише до нашого пікселя, але і між собою. Щоб показати принцип роботи даного алгоритму достатнім буде детальний аналіз лише однієї ітерації. Аналогічно, як і в другому алгоритмі, тут необхідно встановити максимальне відхилення. Нехай воно є рівним "15". Покажемо спочатку на рис. 9 вхідне зображення, інвертовану різницю за абсолютною величиною між вихідним та вхідним файлами, а потім уже результат обробки, який візуально майже не відрізняється від вхідного файлу.

Найбільш цікавим є той факт, що даний алгоритм майже нічого не спотворює, а навпаки – досить акуратно знаходить проблемні пікселі і виправляє. І код цього алгоритму найскладніший, а код першого – відповідно найпростіший. Тобто виходить що, для того, щоб швидко змарнувати зображення великої праці не треба, а виправляти помилки набагато важче.

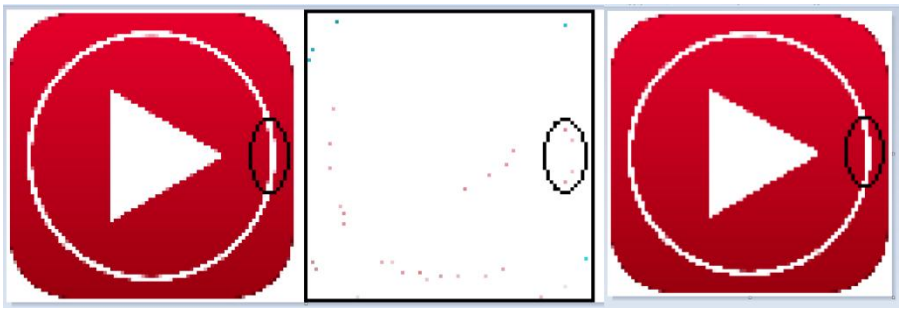


Рисунок 9 – Результати обробки зображення алгоритмом знищення хибних пікселів, що мають групи подібних сусідів

Висновки. Отже, розроблений нами програмний продукт наочно показує як можна реалізувати теорію клітинних автоматів для попіксельної обробки графіки, а саме автоматизованої обробки растрового зображення за допомогою програмування. Хоча кількість методів обробки є обмеженою, але можливим є добавлення будь-якого сучасного відомого алгоритму перетворення графіки (наприклад, за допомогою нашої програми у навчальному процесі можна показувати результати всіх обробок зображення, що пропонуються студентам в курсі таких дисциплін: "Стеганографія", "Обробка цифрових сигналів та зображень" тощо). І це дає можливість далі продовжувати роботу в даному напрямку. Щодо використання розробленого продукту у не навчальних цілях, то це може бути, наприклад, таке перетворення зображення, яке підвищує його якість: знімає розмитість, видаляє зайве (поодинокі шуми – пікселі).

Поставлені проблеми були повністю виконані і є значна перспектива в розширенні зробленого програмного продукту, що вимагає нових досліджень.

1. Архангельский А.Я. Программирование в C++ Builder / А.Я. Архангельский. – М. : ООО "Бином-Пресс", 2010. – 896 с.
2. Архангельский А.Я. Язык C++ в C++ Builder. Справочное и методическое пособие / А.Я. Архангельский. – М. : ООО "Бином Пресс", 2007. – 1012 с.
3. Велъдер С.Э. О верификации простых автоматных программ на основе метода Model Checking / С.Э. Велъдер, А.А. Шалыто // Программные и аппаратные средства. – 2007. – № 3. – С. 27-38.
4. Капітонова Ю.В. Основи дискретної математики / Ю.В. Капітонова, С.Л. Кривий, О.А. Летичевський та ін. – К. : "Наукова думка", 2002. – 580 с.
5. Карпов Ю.Г. Теория автоматов / Ю.Г. Карпов. – СПб. : "Питер", 2002. – 206 с.
6. Культин Н.Б. C++ Builder в задачах и примерах / Н.Б. Культин. – СПб. : "БХВ-Петербург", 2005. – 336 с.
7. Пахомов Б.И. Самоучитель C/C++ и C++ Builder 2007 / Б.И. Пахомов. – СПб. : "БХВ Петербург", 2008. – 672 с.
8. Поликарпова Н.И. Автоматное программирование / Н.И. Поликарпова, А.А. Шалыто. – СПб. : "Питер", 2009. – 167 с.
9. Понимание ООП в JavaScript [Електронний ресурс]. – Режим доступу : http://habrahabr.ru/post/-153365.
10. C++ Community [Електронний ресурс]. – Режим доступу : http://www.cplusplus.com.
11. UniMod: Инструмент для построения схем автоматов в формате UML и генерации по ним исходного кода [Електронний ресурс]. – Режим доступу : http://unimod.sourceforge.net.