**ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ОЛЕСЯ ГОНЧАРА**

Кафедра електронних обчислювальних машин

**КУРСОВА РОБОТА**

з дисципліни «Системне програмування»

Студента II курсу групи КІ-21-2

Спеціальності 123 – «Комп’ютерна інженерія»

Шейко Р.О

(прізвище та ініціали)

Керівник:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів

Національна шкала

Члени комісії:

Пономарьов І.В.

(підпис)

(прізвище та ініціали)

Гниленко О.Б.

(підпис)

(підпис)

(прізвище та ініціали)

(прізвище та ініціали)

Мазурик С.В.

м. Дніпро

2023

# **ЗАВДАННЯ НА КУРСОВУ РОБОТУ**

**Мета:** вивчення принципів побудови системних програм, засвоєння основних засобів та методів конструювання ефективних програм, реалізації алгоритмів обробки інформації різних типів даних та набутті практичних навичок програмування на платформі .NET на прикладі розробки програмного забезпечення обробки зображень.

**Постановка задачі:** у роботі потрібно застосовувати сучасні технології програмування, з використанням методів об'єктно-орієнтованого програмування (ООП):

- створювати класи;

- ієрархію класів;

- розробляти похідні класи.

Окрім цього, використовувати різноманітні структури бібліотеки базових класів Framework Class Library (FCL), зокрема класи Graphics, Image та Bitmap, призначені для роботи з графікою та зображеннями.

Методи та властивості цих класів дозволяють завантаження, збереження і вивід зображень на монітор та принтер, отримання відомості про зображення: формат файлу, розмір (ширину і висоту), число кольорів, роздільну здатність. Крім того є багато методів обробки зображень: обертання, нахил та інші маніпуляції, вписування в прямокутник, малювання на зображенні, створення зображення-тіні. При створенні додатків необхідно розробляти зручний та зрозумілий для користувача графічний інтерфейс. Додавати різні елементи керування такі, як меню, дочірні форми, користувальницькі та стандартні діалогові вікна. Задавати властивості елементів керування та обробляти задані для них події. Виконувати читання даних з файлів та запис результатів обробки у файли, з використанням потоків, класів вводу-виводу

**При виконанні завдання необхідно:**

Створити на мові програмування C# багатовіконний Windows додаток, в меню якого мають бути наступні пункти:

1. Пункт меню Файл має містити такі підпункти:

1) Створити – створення нового дочірнього вікна.

2) Відкрити – відкриття файлів із зображеннями ( формат файлів \*.bmp, \*.jpg, \*.png, \*.gif, \*.tiff).

3) Зберегти – збереження зображення в тому самому форматі.

4) Зберегти як… - збереження зображення в одному з наступних форматів \*.bmp, \*.jpg, \*.png, \*.gif, \*.tiff

5) Закрити – закриття дочірнього вікна. Перед закриттям дочірнього вікна потрібно вивести запит про необхідність збереження змін у файл, якщо вони відбувались.

6) Вихід – закриття всього додатку. Перед закриттям додатку необхідно вивести запит про збереження змін в усіх відкритих файлах, в яких вони відбувалися.

*2.* Пункт меню Інформація має виводити докладні відомості про зображення, що відкрите в активному дочірньому вікні, а саме – ім’я файлу, повний шлях до файлу, формат файлу, розміри в пікселях – висоту та ширину, вертикальну та горизонтальну роздільні здатності (в точках на сантиметр), фізичні розміри в сантиметрах, використаний формат пікселів, використання біта або байта прозорості, число біт на піксель.

*3*. Пункт меню Завдання містить 2 підпункти по кількості індивідуальних завдань, при виборі яких запускається обробка. Використати вікна діалогу для вибору і завдання необхідних параметрів, дочірні вікна для наочного показу результатів обробки. Передбачити використання обробки виняткових ситуацій при введенні користувачем не вірних даних та при роботі з файлами.

Завдання 1. (Варіант 10) Додайте в програму інструмент лупа для збільшення і зменшення зображення. Лупа збільшує зображення при натисканні лівою кнопкою миші, лупа зменшує зображення при натисканні правою кнопкою миші. При виборі цього інструменту через меню або панель інструментів повинен змінюватися курсор миші при знаходженні над клієнтської областю вікна. Кнопка на панелі інструментів і пункт меню повинні бути позначені при виборі відповідного інструменту.

Завдання 2. (Варіант 20) Підсумовування двох зображень або константи і зображення. Функція imadd (X, Y, Z) підсумовує кожен елемент масиву X з відповідним елементом масиву Y і повертає суму відповідних елементів в результуючий масив Z. X і Y представляють собою масиви чисел із плаваючою комою однакового розміру і однакового формату представлення даних. Результуючий масив Z має той же розмір і формат представлення даних, що і Y, коли Y скаляр формату double. В іншому випадку розмірність і формат представлення даних результуючого масиву Z збігається з масивом X. Коли X і Y є масиви цілих чисел і елементи результуючого масиву перевищують допустимий діапазон, то вони скорочуються або округлюються. Продемонструйте в програмі застосування цієї функції.

**ЗМІСТ**

[**ЗАВДАННЯ НА КУРСОВУ РОБОТУ** 2](#_Toc135132067)

[**ВСТУП** 7](#_Toc135132068)

[**ЗБІР ТА АНАЛІЗ ВИМОГ** 8](#_Toc135132069)

[**1.1.** **Умова завдання** 8](#_Toc135132070)

[**1.2.** **Збір та аналіз вимог** 8](#_Toc135132071)

[**2.** **РЕАЛІЗАЦІЯ** 10](#_Toc135132072)

[**2.1.** **Опис логічної структури** 10](#_Toc135132073)

[**2.2.** **Крок 1. Створення класу FormGeneral** 12](#_Toc135132074)

[*2.2.1.* *Опис класу та його властивостей* 12](#_Toc135132079)

[*2.2.2.* *Конструктор класу* ***public FormGeneral().*** *Опис алгоритму* 12](#_Toc135132080)

[*2.2.3.* *Метод* ***private void CreateToolStripMenuItem\_Click()****. Опис алгоритму* 13](#_Toc135132081)

[*2.2.4.* *Метод* ***private void OpenToolStripMenuItem\_Click()****. Опис алгоритму* 14](#_Toc135132082)

[*2.2.5.* *Метод* ***private void SaveToolStripMenuItem\_Click()****. Опис алгоритму* 15](#_Toc135132083)

[*2.2.6.* *Метод* ***private void SaveAsToolStripMenuItem\_Click()****. Опис алгоритму* 16](#_Toc135132084)

[*2.2.7.* *Метод* ***private void CloseToolStripMenuItem\_Click()****. Опис алгоритму* 17](#_Toc135132085)

[*2.2.8.* *Метод* ***private void ExitToolStripMenuItem\_Click().*** *Опис алгоритму* 18](#_Toc135132086)

[*2.2.9.* *Метод* ***private void GetAnswerToSaveAndSave().*** *Опис алгоритму* 18](#_Toc135132087)

[*2.2.10.* *Метод* ***private void Save().*** *Опис алгоритму* 19](#_Toc135132088)

[*2.2.11.* *Метод* ***private void InformationToolStripMenuItem\_Click().*** *Опис алгоритму* 19](#_Toc135132089)

[*2.2.12.* *Метод* ***private void Task1ToolStripMenuItem\_Click().*** *Опис алгоритму* 20](#_Toc135132090)

[*2.2.13.* *Метод* ***private void Task2ToolStripMenuItem\_Click()****. Опис алгоритму* 21](#_Toc135132091)

[**2.3.** **Крок 2. Створення класу FormChild** 22](#_Toc135132099)

[*2.3.1.* *Опис класу та його властивостей* 22](#_Toc135132100)

[*2.3.2.* *Конструктор класу* ***public FormChild ().*** *Опис алгоритму* 22](#_Toc135132101)

[*2.3.3.* *Метод* ***public void FormChild\_Paint()****. Опис алгоритму* 23](#_Toc135132102)

[*2.3.4.* *Метод* ***public Bitmap GetChangedImage()****. Опис алгоритму* 24](#_Toc135132103)

[*2.3.5.* *Метод* ***private byte GetNumPixel().*** *Опис алгоритму* 25](#_Toc135132104)

[*2.3.6.* *Метод* ***public void UploadImageToBuffer().*** *Опис алгоритму* 26](#_Toc135132105)

[*2.3.7.* *Метод* ***private void FormChild\_Resize().*** *Опис алгоритму* 26](#_Toc135132106)

[*2.3.8.* *Метод* ***private void FormChild\_Click().*** *Опис алгоритму* 26](#_Toc135132107)

[*2.3.9.* *Метод* ***private void ChangePoint\_NumericUpDown\_ValueChanged().*** *Опис алгоритму* 27](#_Toc135132108)

[**2.4.** **Крок 3. Створення класу Const та підкласу Message** 28](#_Toc135132114)

[**3.** **ТЕСТУВАННЯ** 29](#_Toc135132123)

[**3.1 Крок 1. Тестування меню програми(протестуємо)** 29](#_Toc135132127)

[**3.2 Крок 2. Тестування індивідуального завдання** 33](#_Toc135132132)

[**Висновок** 36](#_Toc135132133)

[**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ** 37](#_Toc135132134)

[**ДОДАТКИ** 38](#_Toc135132135)

# **ВСТУП**

В ХХI столітті ІТ сфера ще більш, і більш пронизує наше повсякденне життя. Просинаємось від додатку «Будильник», відслідковуємо погоду з «Погодою», їдемо на роботу з додатком «Google maps». Але, ми всі ті ж люди, що і 2000 тисяч років тому потребуємо соціалізації, спілкування з рідними й важливими для нас людьми. Ми живимо в моменті, й іноді нам потрібно зберігти моменти, маленькі спалахи й вогінчики нашого життя.

Перегляд фотографій - це виключно популярна форма розваги та спосіб зберігання спогадів. Сучасні технології дозволяють легко зберегти та організувати велику кількість зображень, а також поділитися ними з іншими людьми, використовуючи соціальні мережі та інші інтернет-ресурси.

В цій роботі висвітлено дослідження в області програмування, й стосується вивченню та засвоєнню навичок з мови програмування «С#», оволодіння принципами розробки системних програм, засвоєння основних методів створення ефективних програм та реалізації алгоритмів для обробки різних типів даних. Було отримано практичні навички програмування на платформі .NET, на прикладі розробки програмного забезпечення для обробки зображень.

Мета курсової роботи концентрована на аналізі засобів мови С# для реалізації задачі, , що пов’язана зі створенням програмного забезпечення для обробки зображень. В ПЗ повинні бути реалізовані такі функції: відкриття, збереження, наближення, зміни кольору, створення додаткових вікон для перегляду зображень.

Для досягнення мети роботи треба виконати наступні задачі:

1. Вивчення теоретичної частини за допомогою навчальних матеріалів та книг.
2. Навчитися оброблювати зображення за допомогою мови програмування.
3. Виконати наведені задачі.
4. Додати описання використаних методів та отриманих даних.
5. Зробити висновки щодо виконаної роботи, закріпити матеріал.

Об’єктом дослідження є процес розробки програмного забезпечення.

# **ЗБІР ТА АНАЛІЗ ВИМОГ**

* 1. **Умова завдання**

Скласти програмне забезпечення для перегляду картинок для дослідження мови програмування «С#». Так, для реалізації програмного забепечення потрібно реалізувати такі пункти меню:

* + - 1. Пункт меню Файл з підпунктами: створити, відкрити, зберегти, зберегти як, закрити і вихід. Вони будуть виконувати відповідні дії з зображеннями до назви підпунктів.
      2. Пункт меню Інформація, що має виводити докладні відомості про зображення, що відкрите в активному дочірньому вікні.
      3. Пункт меню Завдання містить 2 підпункти Завдання 1 та Завдання 2. Вони відповідають вибранним варіантом завдання. Завдання 1 – лупа, що буде збільшувати зображення при натисканні. Завдання 2 – підсумовування константи і зображення. Малюнок змінює колір відповідно до числа, що було введено.
  1. **Збір та аналіз вимог**

ПЗ «Перегляд зображень» повинна надавати можливість користувачу:

* + - Створювати дочірні вікна;
    - Відкривати зображення;
    - Закривати зображення;
    - Зберігати зображення у тому ж файлі;
    - Зберігати зображення у іншому файлі;
    - Отримувати всю інформацію про зображення;
    - Приближувати зображення;
    - Змінювати колір картинок;
  1. **Моделювання вимог**

На рис. 1.1 наведено моделювання вимог користувача до ПЗ «Перегляд зображень» за допомогою діаграми варіантів використання (use-case diagram). Оскільки кожній людині, яка буде користуватись програмою перегляд зображень буде надано однаковий функціонал, то у нашому випадку буде лише один актор. Він може виконувати дії (прецеденти): переглядати, приближувати чи отримувати інформацію(зображення), закривати, відкрити й змінювати зображення й після цього його зберігати(обов’язково перед закриттям).

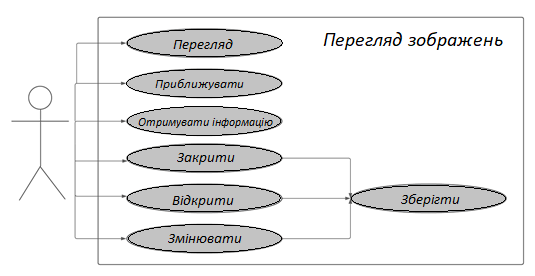


Рисунок 1.1. – Моделювання вимог користувача за допомогою діаграми use-case

Кожен прецедент – це окрема функція в системі, яка складається з послідовності виконуваних дій. Поведінку не тільки системи, але й окремої функції можна змоделювати за допомогою діаграми активності (Activity diagram). Зовні вона нагадує блок-схему, в якій замість блоків «Початок» та «Кінець» використовують спеціальні символи – чорний круг та «котяче око». Ця діаграма підтримує малювання оператору switch (множинний вибір) та розпаралелювання потоків (використовується під час паралельних та розподілених обчислень).

1. **РЕАЛІЗАЦІЯ**
   1. **Опис логічної структури**

Програма складається з трьох програмних модулів - файлу **FormGeneral.cs,** файлу **FormChild.cs** тафайлу **Const.cs**. Вони і є классами.

До класу **FormGeneral** входять такі властивості:

* public Bitmap ImageBuffer - буфер для зображення;
* public string ImagePath - путь до зображення;
* public bool IsZoomWorking – статус приближення;
* public bool IsNumericChangerEnable – статус змінителя зображення;
* private int \_nextFormNumber – порядковий номер наступного числа;

До класу **FormGeneral** входять такі методи:

* + public FormGeneral() – конструктор класу;
  + private void CreateToolStripMenuItem\_Click() – створення нового дочірнього вікна;
  + private void OpenToolStripMenuItem\_Click() – відкриття нового зображення;
  + private void SaveToolStripMenuItem\_Click() – збереження зображення у тому ж файлі;
  + private void SaveAsToolStripMenuItem\_Click() – збереження зображення у іншому файлі;
  + private void CloseToolStripMenuItem\_Click() – закриття дочірнього вікна;
  + private void ExitToolStripMenuItem\_Click() – закриття вікон та додатку;
  + private void GetAnswerToSaveAndSave() – запит у користувача на запис зображення;
  + private void Save() – збереження зображення;
  + private void InformationToolStripMenuItem\_Click() – отримання інформації про зображення;
  + private void Task1ToolStripMenuItem\_Click() – активація режиму лупи;
  + private void Task2ToolStripMenuItem\_Click() – активація режиму змінення зображення;

До класу **FormChild** входять такі властивості:

* public Bitmap ImageBuffer – буфер зображення;
* public ImageFormat ImageFormat – буфер формату зображення;
* public string ImagePath – буфер шляху;
* public bool IsChanged – статус зміни зображення;
* public bool IsZoomed – статус приближення;
* private FormGeneral \_parent – посилання на головну форму;
* private int coefficient – коефіціент для приближення;

До класу **FormChild** входять такі методи:

* + public **FormChild** () – конструктор класу;
  + public void FormChild\_Paint() – прорисовування графіки;
  + public Bitmap GetChangedImage() – повертає зображення;
  + private byte GetNumPixel() – повертає номер для Pixel;
  + public void UploadImageToBuffer() – підгрузка данних з головної форми у дочірню;
  + private void FormChild\_Resize() – прорисовування при зміні розміру;
  + private void FormChild\_Click() – виконує приближення;
  + private void ChangePoint\_NumericUpDown\_ValueChanged() – зміна статусу IsChanged;

До класу **Const** входять такі константи:

* public const string FilesFilter – фільтр файлів;
* public const string ImageFileName – дефолтна назва файлу;
* public const string ProgramName – назва програми;

До класу підкласу **Messages** класа **Const** входять такі повідомлення:

* public const string Attention – увага;
* public const string CreateFormToViewImage – створити форму для відображення зображення;
* public const string ImageInThisFile – зберігати зображення або ні;
* public const string DownloadImageToBuffer – скачувати в буфер картинку;
* public const string FileNotWasChanged – файл не був змінений;
* public const string ActiveFormIsNull – активної форми не існує;
* public const string ImageBuffesIsNull – буфер зображень пустий;
* public const string IsNullOrWhiteSpace – шлях до файлу пустий або не існує;

## **Крок 1. Створення класу FormGeneral**



### *Опис класу та його властивостей*

У **додатку А** знаходиться код до класу **FormGeneral.cs**.

Створюємо властивість для збереження й передачі до дочірньої форми зображення – public Bitmap ImageBuffer { get; private set; }.

Створюємо властивість для збереження й передачі до дочірньої форми шлях – public string ImagePath { get; private set; }.

Створюємо властивість для відстеження роботи приближення – public bool IsZoomWorking { get; private set; }.

Створюємо властивість для відстеження роботи зміни зображення – public bool IsZoomWorking { get; private set; }.

Створюємо властивість для відстеження номера форм – private int \_nextFormNumber { get; set; }.

* + 1. *Конструктор класу* ***public FormGeneral().*** *Опис алгоритму*

Конструктор форми **FormGeneral** викликається спочатку створення об`єкту і в ньому ми ініціалізуємо форму, через InitializeComponent() – передвстановленний метод класу Form.

Далі привласнюємо до властивостей класу дефолтні значення:

* + ImageBuffer = null;
  + ImagePath = string.Empty;
  + \_nextFormNumber = 1;
  + IsZoomWorking = false.

Привласнюємо назву до програми курсача за допомогою звернення до поля Text. Надалі дозволяємо формі бути контейнером для MDI форм через поле IsMdiContainer. Встановлюємо для об’єкту OpenFileDialogWindow дефолтну назву файлу звертаючись до його поля FileName. Далі привласнюємо у фільтр цього об’єкту з класу Const константи FilesFilter за допомогою звернення до поля Filter.

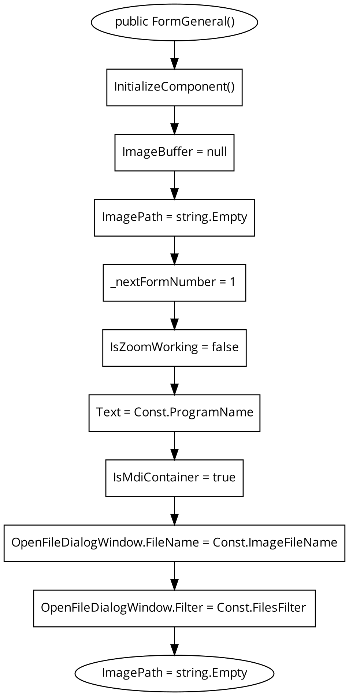


Рис 2.1. – Блок схема конструктору FormGeneral

* + 1. *Метод* ***private void CreateToolStripMenuItem\_Click()****. Опис алгоритму*

Цей метод існує для створення й відображення форми. Також він робить першочергове налаштування.

Спочатку відтворення методу ми створюємо новий екземпляр дочірньої форми за допомогою створення об’екту new FormChild(this, "Редактор" + \_nextFormNumber++). Далі ми показуємо форму за допомогою методу Show(). Завантажуємо картинку до буферу за допомогою методу UploadImageToBuffer() й далі, якщо на головній формі включений режим змін картинки, то включаємо його й у дочірній.

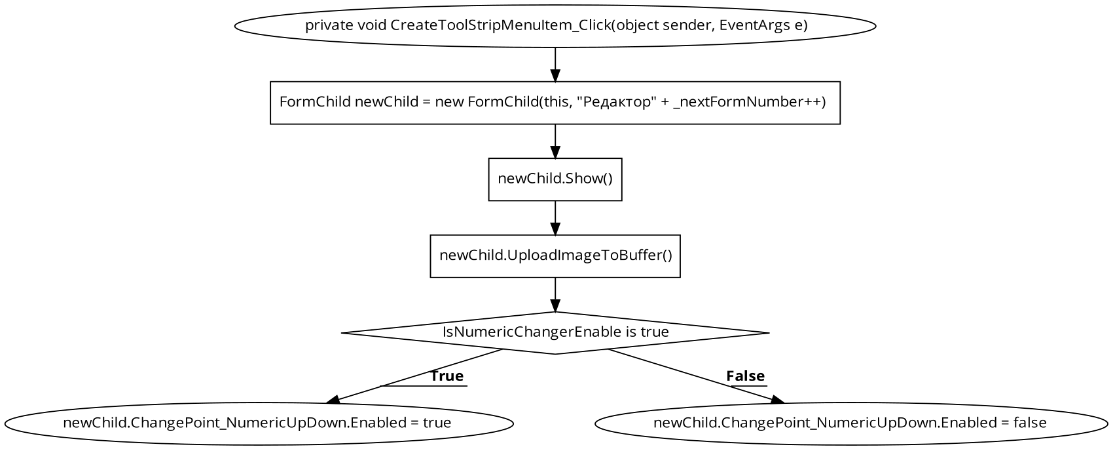


Рис 2.2. – Блок схема методу CreateToolStripMenuItem\_Click()

* + 1. *Метод* ***private void OpenToolStripMenuItem\_Click()****. Опис алгоритму*

Цей метод існує для відкривання файлів.

Спочатку виклику метода відкривається діалог відкриття файлу. Якщо він закінчується позитивно, то ми продовжуємо виконання методу, але в іншому разі ми виконання методу завершується. Далі ми записуємо обрану назву файлу у ImagePath. Далі ми відкриваємо файл через конструкцію using та завантажуємо. Якщо викликалась одна з помилок, то виводиться помилка на екран та виконання методу завершується.

Наступним кроком ми визначаємо, чи є в додатку активна форма. Якщо її нема, то пропонуємо створити дочірнє вікно й у разі позитивної відповіді викликаємо функцію створення CreateToolStripMenuItem\_Click(). В іншому разі обираємо активну дочірню форму й туди підгружаємо картинку за допомогою метода UploadImageToBuffer() та перерисовуємо графіку Invalidate(). Очищуємо властивості – буфери: ImageBuffer і ImagePath.

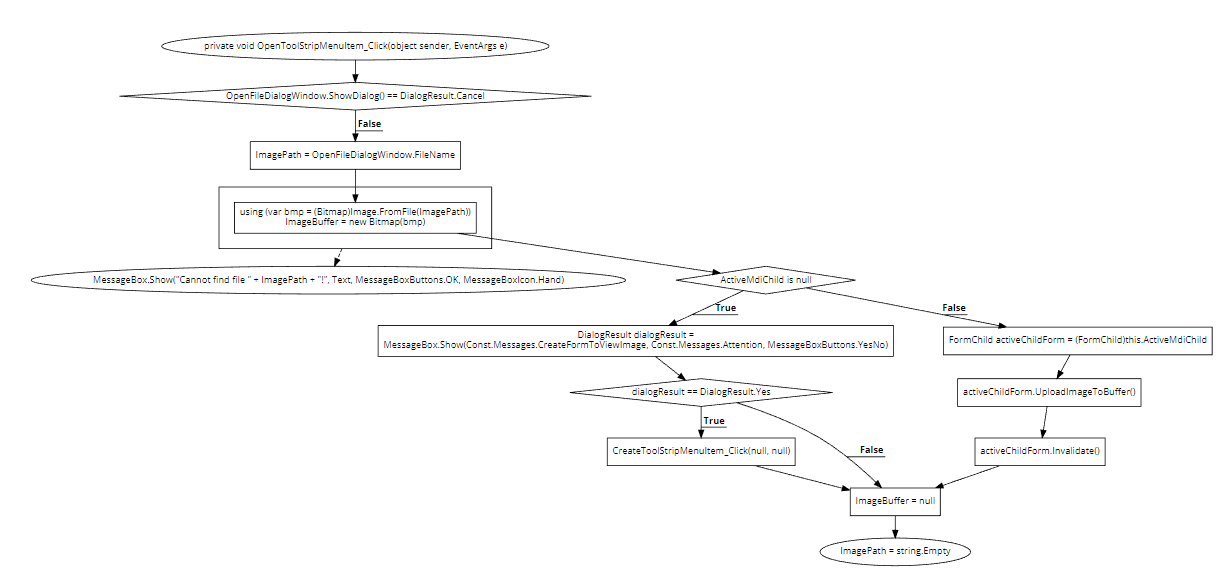


Рис 2.3. – Блок схема методу OpenToolStripMenuItem\_Click()

* + 1. *Метод* ***private void SaveToolStripMenuItem\_Click()****. Опис алгоритму*

Цей метод існує для збереження файлів у тому ж файлі.

Спочатку перевіряємо, чи існує активна форма. Якщо її не існує, то метод припиняє своє виконання. Далі отримуємо активну форму. Наступним кроком запитуємо у користувача та якщо відповідь позитивна, переходимо у функцію Save().

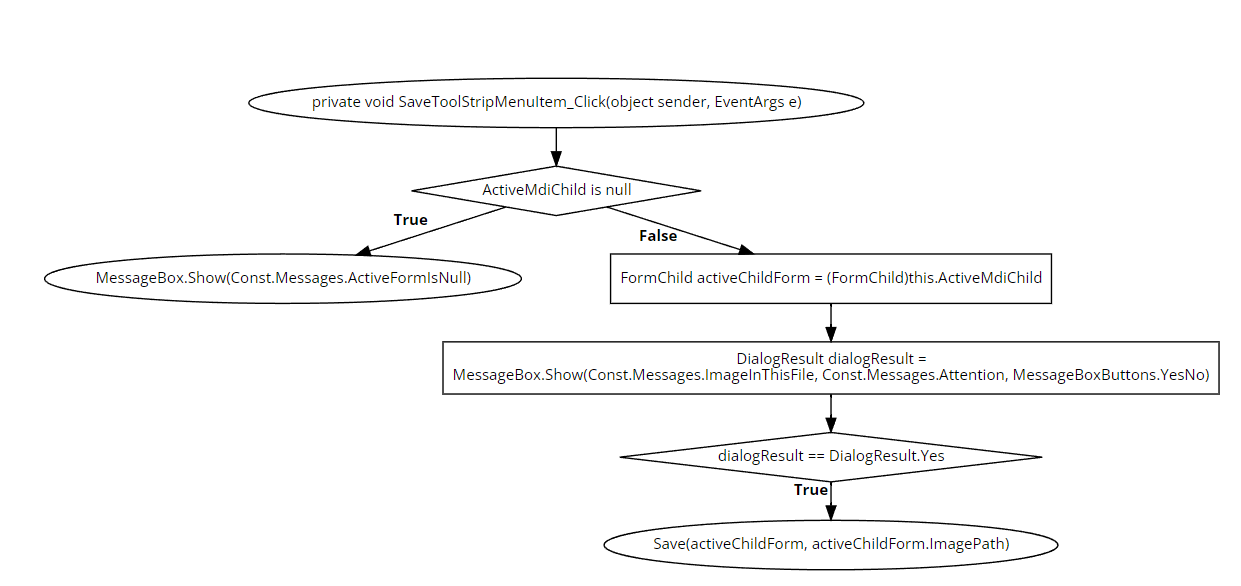


Рис 2.4. – Блок схема методу SaveToolStripMenuItem\_Click()

* + 1. *Метод* ***private void SaveAsToolStripMenuItem\_Click()****. Опис алгоритму*

Цей метод існує для збереження файлів у обраному користувачем файлу.

Спочатку перевіряємо, чи існує активна форма. Якщо її не існує, то метод припиняє своє виконання. Далі створюється об’єкт SaveFileDialog й встановлюємо фільтр Const.FilesFilter. Наступним кроком отримуємо активну форму й якщо користувач натиснув Ok, то переходимо у функцію Save().

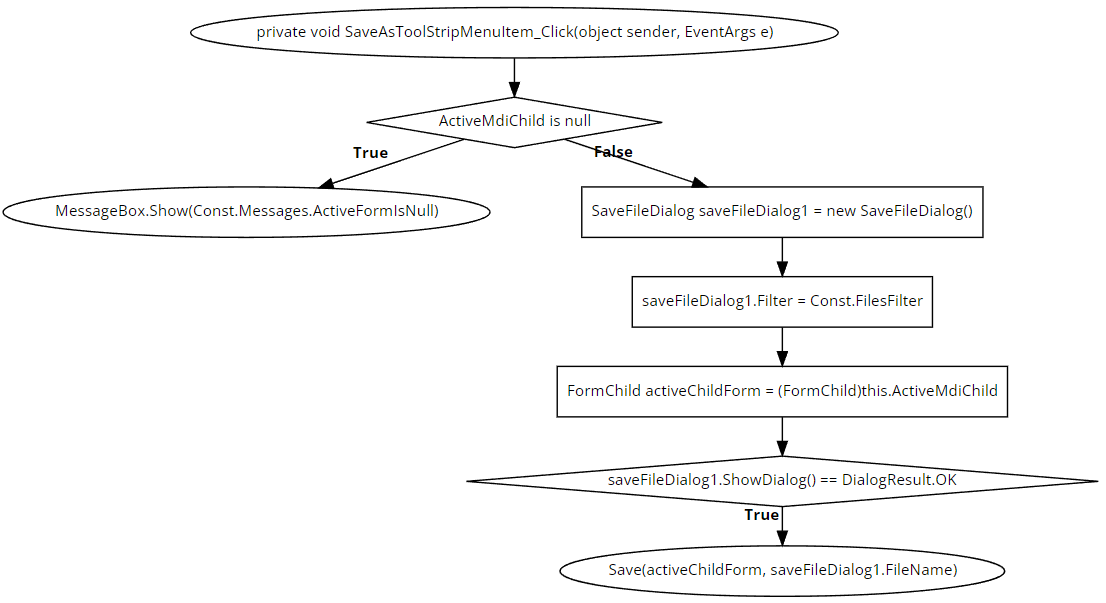


Рис 2.5. – Блок схема методу SaveAsToolStripMenuItem\_Click()

* + 1. *Метод* ***private void CloseToolStripMenuItem\_Click()****. Опис алгоритму*

Цей метод існує для закриття обранної користувачем форми.

Спочатку перевіряємо, чи існує активна форма. Якщо її не існує, то метод припиняє своє виконання. Отримуємо активну форму. Викликаємо функцію GetAnswerToSaveAndSave() й потім закриваємо активну форму через метод Close().

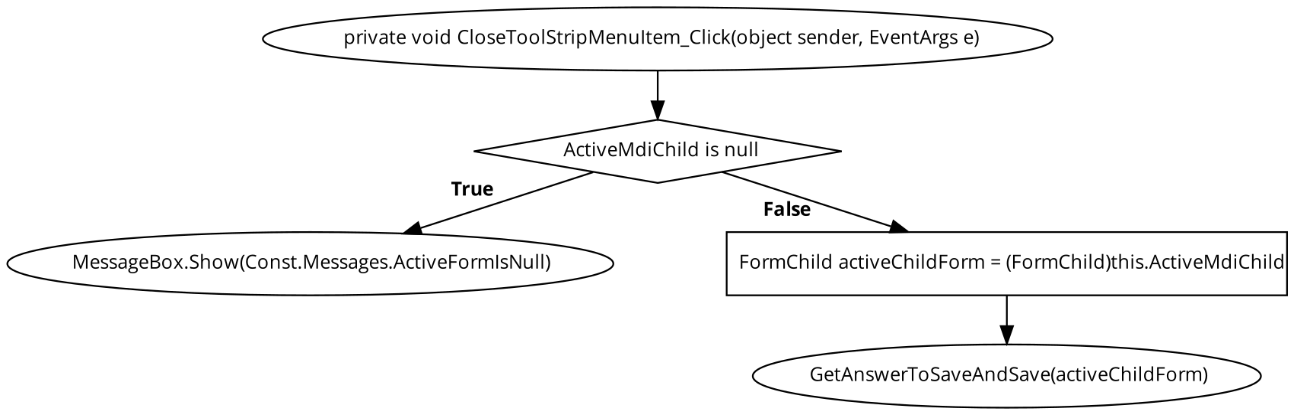


Рис 2.6. – Блок схема методу CloseToolStripMenuItem \_Click()

* + 1. *Метод* ***private void ExitToolStripMenuItem\_Click().*** *Опис алгоритму*

Цей метод існує для повного закриття програмного забезпечення.

Спочатку перевіряємо, чи існує активна форма. Якщо її не існує, то закриваємо додаток. В іншому разі запускаємо цикл, що отримує кожну форму додатку й запускаємо в ній метод GetAnswerToSaveAndSave(). Потім закриваємо додаток.

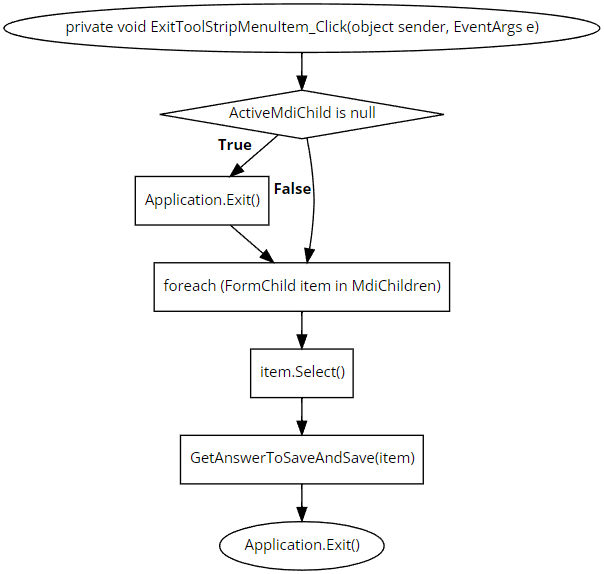


Рис 2.7. – Блок схема методу ExitToolStripMenuItem \_Click()

* + 1. *Метод* ***private void GetAnswerToSaveAndSave().*** *Опис алгоритму*

Цей метод існує для запиту у користувача хоче чи він зберігати зображення. Якщо він відповідає так, то переходимо у метод Save(). В іншому разі якщо файл не був змінений або користувач відповів ні, нічого не відбувається.

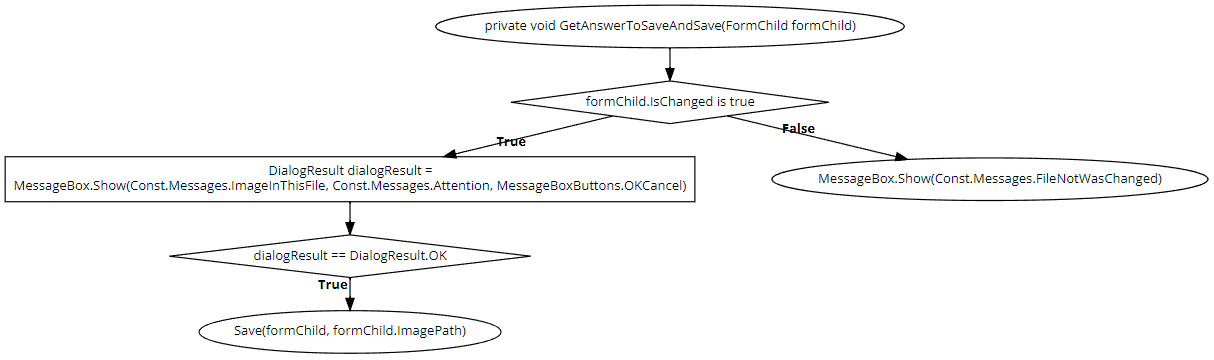


Рис 2.8. – Блок схема методу GetAnswerToSaveAndSave ()

* + 1. *Метод* ***private void Save().*** *Опис алгоритму*

Цей метод існує для збереження файлів.

Спочатку перевіряємо чи пустий буфер картинки. Якщо він пустий, то закінчуємо виконання цього методу. Далі перевіряємо, пустий чи шлях. Якщо він пустий, то закінчуємо виконання цього методу. Далі видаляємо файл й зберігаємо файл. Далі закриваємо форму. Якщо відскачує помилка, то вона виводиться.

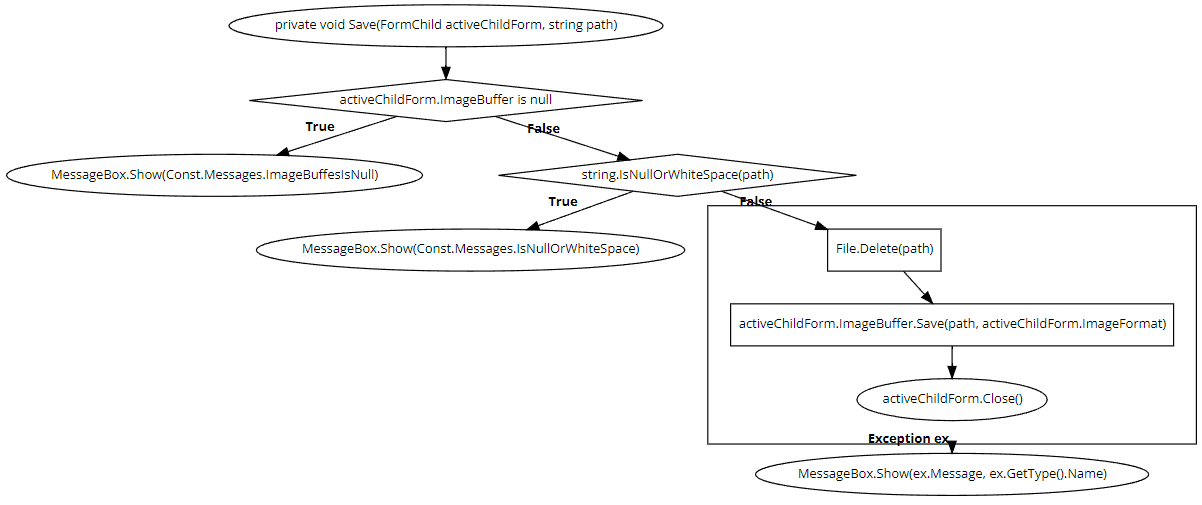


Рис 2.9. – Блок схема методу Save ()

* + 1. *Метод* ***private void InformationToolStripMenuItem\_Click().*** *Опис алгоритму*

Цей метод існує для виводу інформації про зображення.

Спочатку перевіряємо, чи існує активна форма. Якщо її не існує, то закриваємо додаток. Отримуємо активну форму. Далі перевіряємо чи пустий буфер картинки. Якщо він пустий, то закінчуємо виконання цього методу. Оголошуємо змінні: inch, dpiX, dpiY, dpiBase. Отримуємо об’єкт Graphics. Формулюємо text й виводимо в MessageBox.

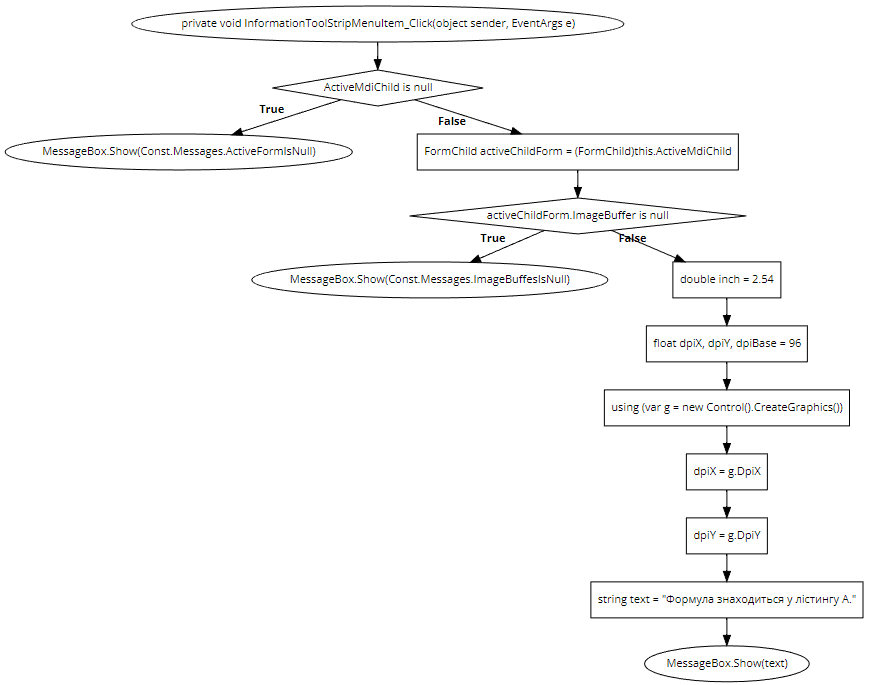


Рис 2.10. – Блок схема методу InformationToolStripMenuItem\_Click()

* + 1. *Метод* ***private void Task1ToolStripMenuItem\_Click().*** *Опис алгоритму*

Цей метод існує для корегування статусу завдання 1.

Спочатку роботи методу ми привласнюємо статусу IsZoomWorking протилежне значення. Далі, в залежності від режиму змінюємо курсор миші.

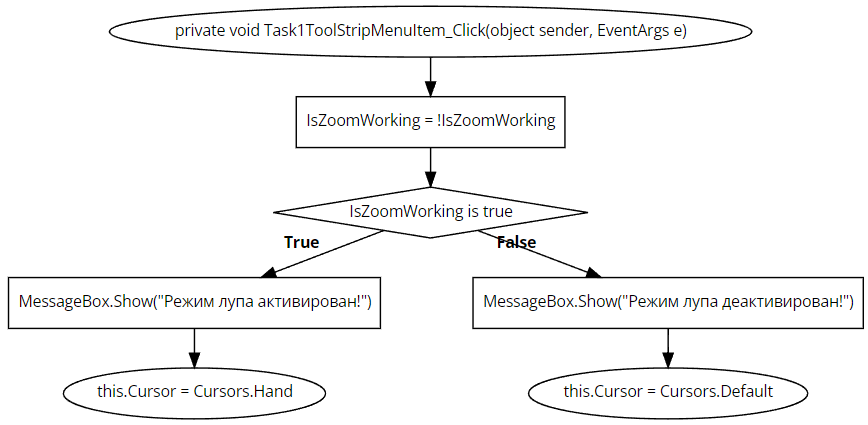


Рис 2.11. – Блок схема методу Task1ToolStripMenuItem \_Click()

* + 1. *Метод* ***private void Task2ToolStripMenuItem\_Click()****. Опис алгоритму*

Цей метод існує для корегування статусу завдання 2.

Спочатку перевіряємо, чи існує активна форма. Якщо її не існує, то метод припиняє своє виконання. Отримуємо активну форму. роботи методу ми привласнюємо статусу IsNumericChangerEnable протилежне значення. Далі, в залежності від режиму виключаємо або включаємо через цикл на усіх формах Changer.

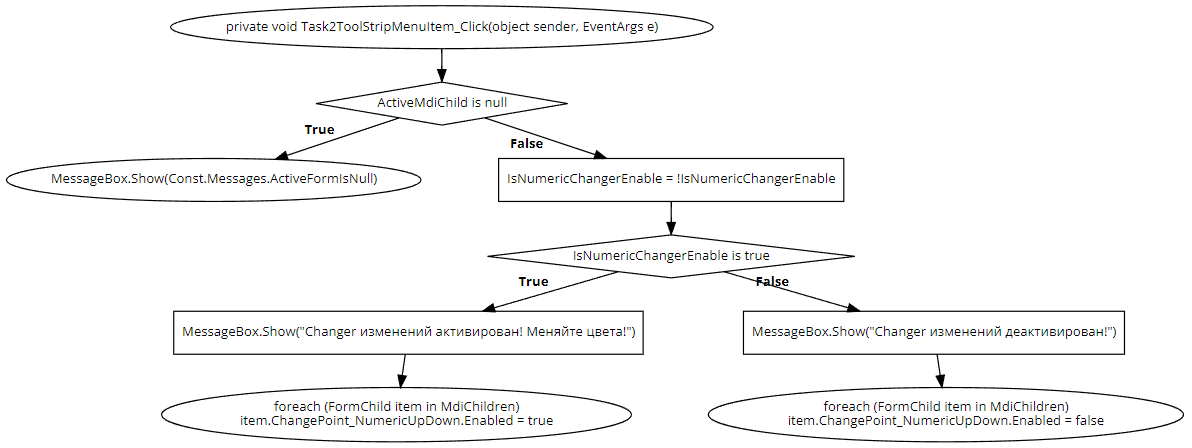


Рис 2.12. – Блок схема методу Task2ToolStripMenuItem \_Click()



## **Крок 2. Створення класу FormChild**

### *Опис класу та його властивостей*

У **додатку Б** знаходиться код до класу **FormChild.cs**.

Створюємо властивість для збереження зображення – public Bitmap ImageBuffer { get; private set; }.

Створюємо властивість для збереження формату зображення – public ImageFormat ImageFormat { get; private set; }.

Створюємо властивість для збереження шляху – public string ImagePath { get; private set; }.

Створюємо властивість для статусу приближення – public bool IsZoomed { get; private set; } .

Створюємо властивість для статусу зміни зображення – public bool IsChanged { get; private set; } .

Створюємо властивість для посилання на головну форму – private FormGeneral \_parent { get; set; }.

Створюємо властивість для посилання на головну форму – private FormGeneral \_parent { get; set; }.

Створюємо властивість для коефіціенту зуму – private int coefficient {get; set; }.

* + 1. *Конструктор класу* ***public FormChild ().*** *Опис алгоритму*

Конструктор форми **FormChild** викликається спочатку створення об`єкту і в ньому ми ініціалізуємо форму, через InitializeComponent() – передвстановленний метод класу Form.

Далі привласнюємо до властивостей класу дефолтні значення:

* + ImageBuffer = null;
  + ImagePath = string.Empty;
  + IsChanged = false;
  + IsZoomed = false;
  + ChangePoint\_NumericUpDown.Enabled = false;
  + \_parent = parent;
  + coefficient = 2;
  + this.MdiParent = parent;
  + this.Text = caption.

Також підписуємось на дві події:

* + this.Paint += FormChild\_Paint;
  + this.Resize += FormChild\_Resize;

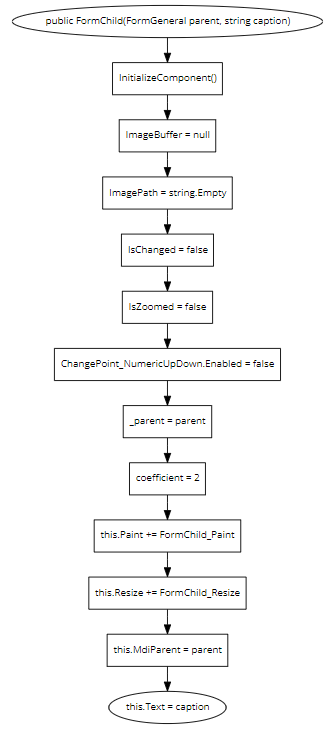


Рис 2.13. – Блок схема конструктору FormChild

* + 1. *Метод* ***public void FormChild\_Paint()****. Опис алгоритму*

Цей метод існує для прорисування малюнків у вікні.

Спочатку перевіряється, чи пустий буфер зображення. Якщо не пустий, то виконання методу продовжується. Далі створюється об’єкт Graphics та оголошується змінна userImage. Якщо зміни відбувалися, то отримуємо змінене зображення. В іншому разі беремо те, що лежить у буфері. Далі, якщо зум виконувався, то відображуємо нормальну картинку, а в іншому разі – приближену. Очищаємо графіку. Далі, якщо у малюнку були зміни, то ми запрошуємо у користувача, чи хоче він підгрузити новий малюнок у буфер. Якщо він дає на це дозвіл, в ImageBuffer буде загружено новий малюнок.

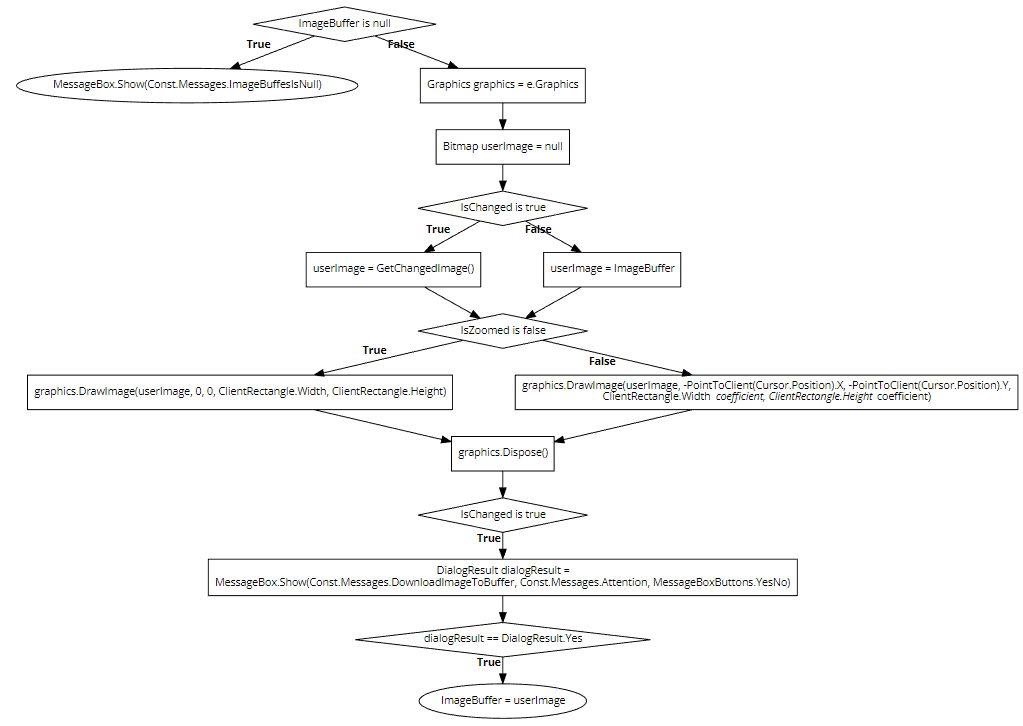


Рис 2.13. – Блок схема методу FormChild\_Paint()

* + 1. *Метод* ***public Bitmap GetChangedImage()****. Опис алгоритму*

Ця функція існує для створення нової картинки. Вона виконує функцію методу *imadd()* з **завдання 2**.

Спочатку ми створюємо змінну, у яку переносимо картинку з буферу. Далі завдяки вкладеному циклу ми перебираємо усі пікселі картинки й створюємо нові.

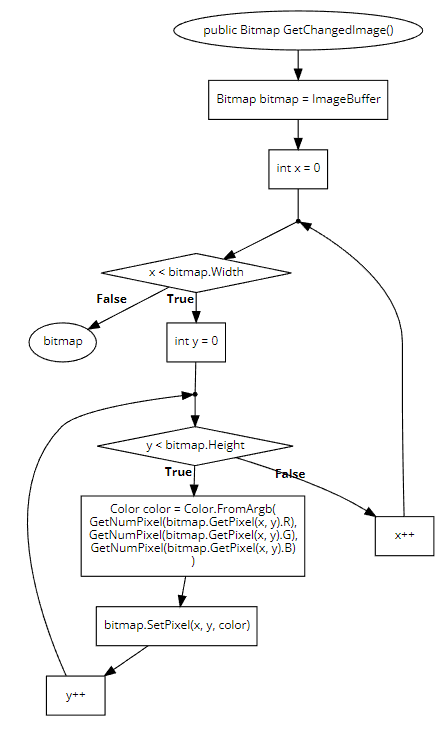


Рис 2.14. – Блок схема методу GetChangedImage()

* + 1. *Метод* ***private byte GetNumPixel().*** *Опис алгоритму*

Ця функція існує для отримання номеру в системі RGB. Тобто, числа від 0 до 255 для зміни пікселі. Якщо у ході конвертації викидується помилка, то ми примусово призначаємо число 254.

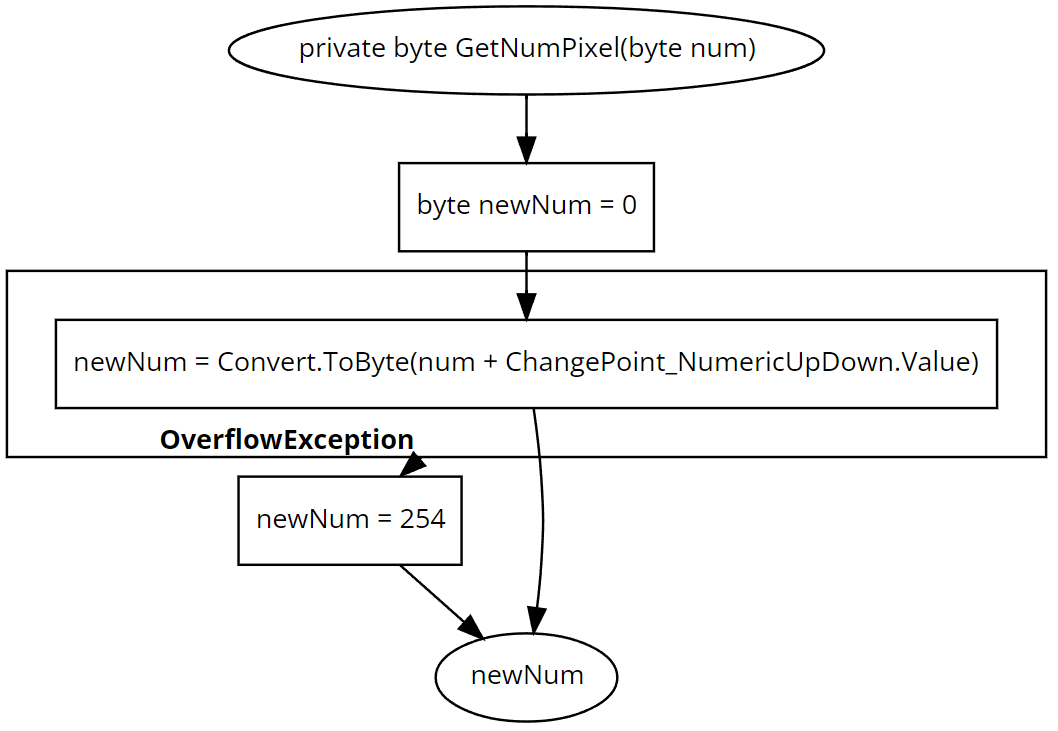


Рис 2.15. – Блок схема методу GetNumPixel ()

* + 1. *Метод* ***public void UploadImageToBuffer().*** *Опис алгоритму*

Цей метод існує для підгрузки картинки та інших даних до буферу цієї форми.

Спочатку перевіряємо, чи пустий буфер зображення у батьківській формі. Якщо пустий, то виконання цього методу завершується. Далі перевіряємо путь картинки. Якщо він теж пустий, то виконання цього методу завершується. Далі у буфер форми підгружаємо малюнок, шлях та формат.

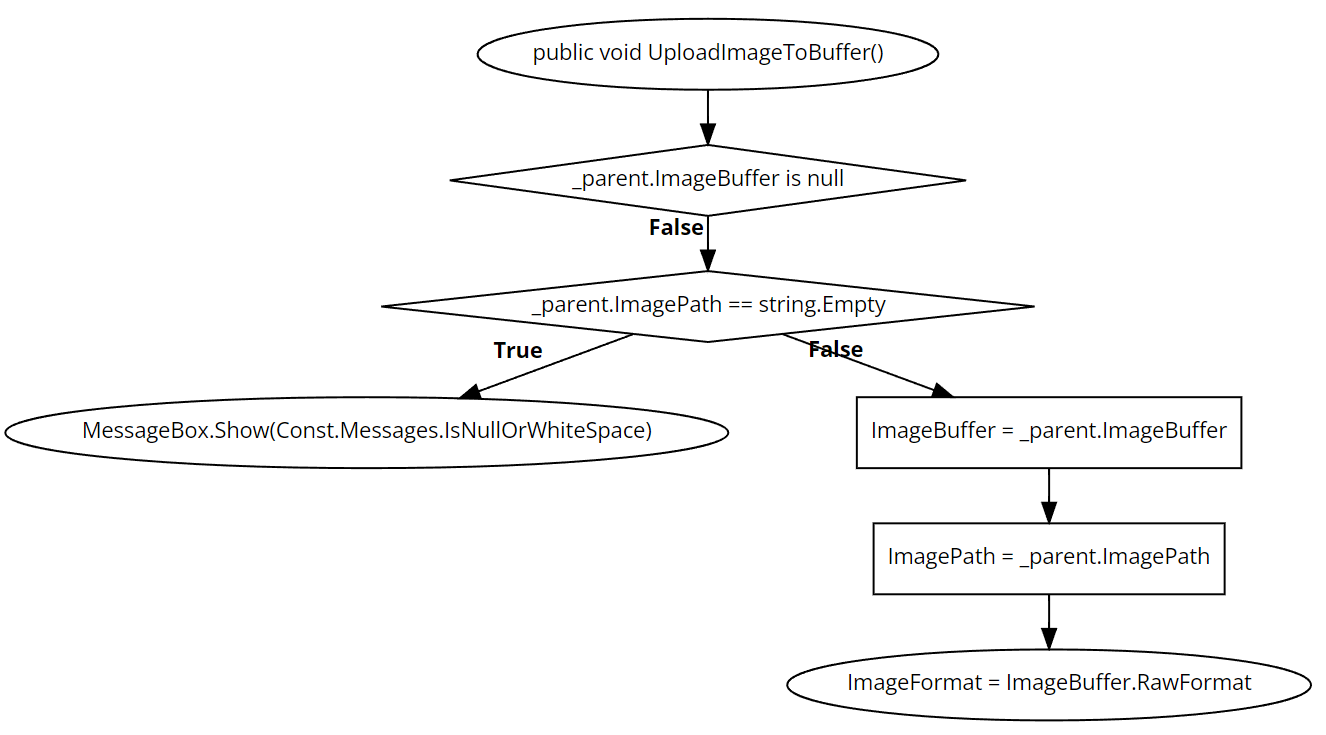


Рис 2.16. – Блок схема методу UploadImageToBuffer ()

* + 1. *Метод* ***private void FormChild\_Resize().*** *Опис алгоритму*

Цей метод викликає Invalidate(), що у свою чергу викликає подію Paint.

* + 1. *Метод* ***private void FormChild\_Click().*** *Опис алгоритму*

Спочатку дії фукнції, перевіряється, чи активне наближення. Якщо так, то далі переключаємо наближення й викликаємо Invalidate().

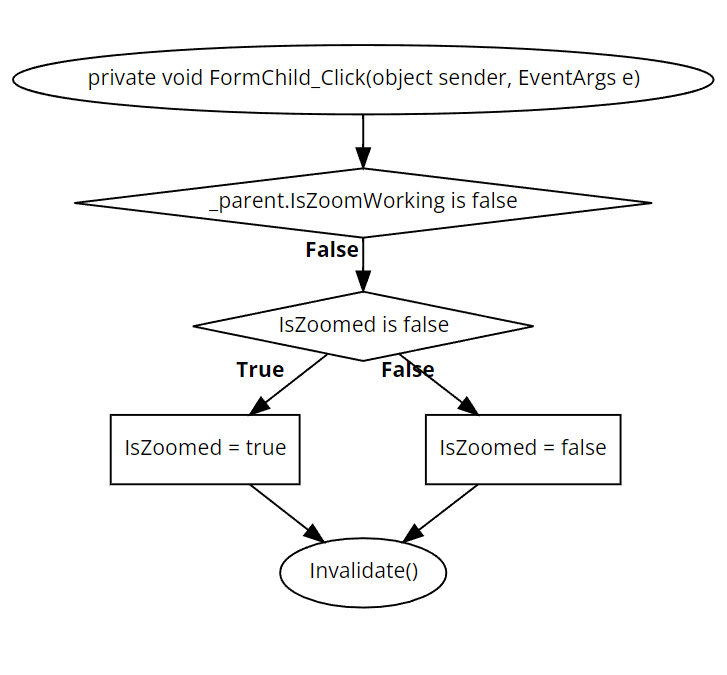


Рис 2.17. – Блок схема методу FormChild\_Click ()

* + 1. *Метод* ***private void ChangePoint\_NumericUpDown\_ValueChanged().*** *Опис алгоритму*

Спочатку дії функції перевіряємо. Якщо змін до цього не було, то IsChanged = true та перемальовуємо вікно викликаючи Invalidate().

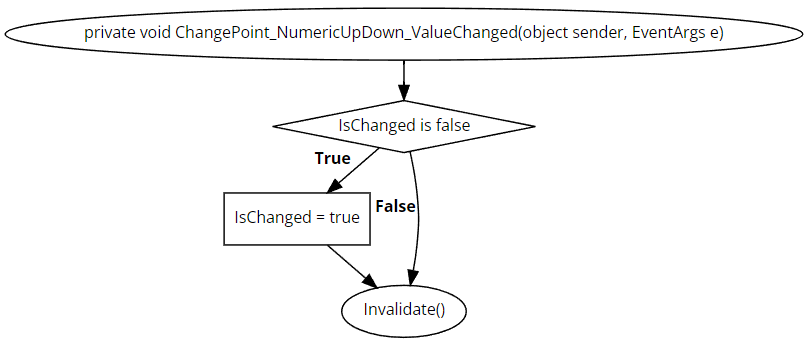


Рис 2.17. – Блок схема методу ChangePoint\_NumericUpDown\_ValueChanged ()



## **Крок 3. Створення класу Const та підкласу Message**








У **додатку В** знаходиться код до класу **Const.cs**.

Клас Const та підклас Message – це класи повністю з константами та різними повідомлення для користувача.

Ось список усіх констант класу Const:

* public const string FilesFilter - фільтр;
* public const string ImageFileName – назва картинки за замовчунням;
* public const string ProgramName – назва проекту;

Та список усіх констант-повідомлень підкласу Message:

* public const string Attention – «Увага»;
* public const string CreateFormToViewImage – про створення форми;
* public const string ImageInThisFile – про збереження у цьому ж файлі;
* public const string DownloadImageToBuffer – про пігрузку картинки у буфер;
* public const string FileNotWasChanged – про те, що файл не був змінений;
* public const string ActiveFormIsNull – про не існуючу активну форму;
* public const string ImageBuffesIsNull – про пустий буфер малюнків;
* public const string IsNullOrWhiteSpace – про пустий простір.

1. **ТЕСТУВАННЯ**

## **3.1 Крок 1. Тестування меню програми(протестуємо)**

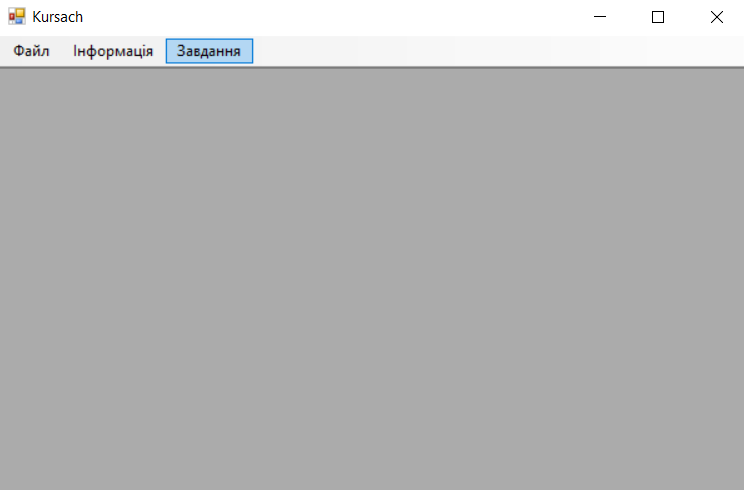


Рис. 3.1 – Меню програми

|  |  |
| --- | --- |
|  |  |
| Рис. 3.2 а – Обираємо пункт меню «Створити» | Рис. 3.2 б – Обираємо пункт меню «Створити» з пустим буфером зображення |

|  |  |
| --- | --- |
|  |  |
| Рис. 3.3 а – Обираємо пункт меню «Відкрити» | Рис. 3.3 б – Обираємо одне з зображень |

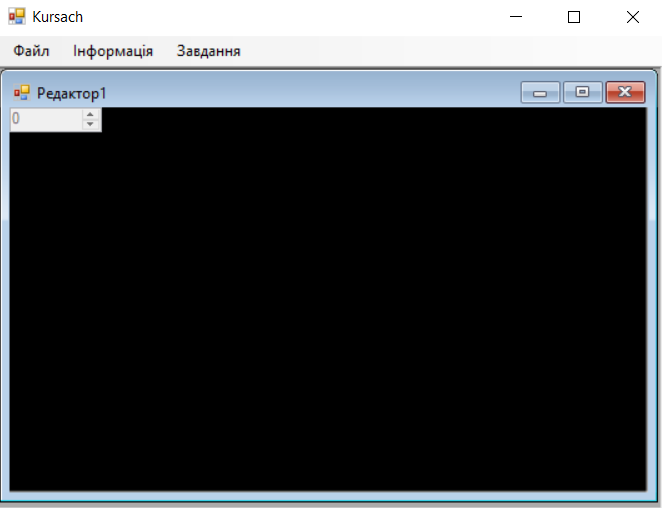


Рис. 3.3 в – Відображення зображення

Також закриємо дочірню форму й просто відкриємо зображення. Після вибору зображення у нас відкриється діалогове вікно з пропозицією:

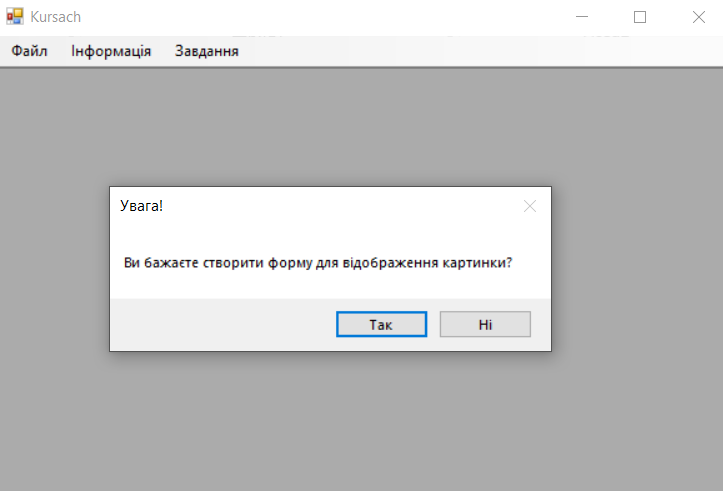


Рис. 3.4 – Відображення пропозиції

Якщо обираємо так, то відобразиться Рис. 3.3 в, а в іншому разі нічого не відбувається.

|  |  |
| --- | --- |
|  |  |
| Рис. 3.5 а – Обираємо пункт меню «Зберегти» | Рис. 3.6 б – Відображення пропозиції зберегти |

Після обрання одно з варіантів дочірня форма закриється.

|  |  |
| --- | --- |
|  |  |
| Рис. 3.6 а – Обираємо пункт меню «Зберегти як…» | Рис. 3.6 б – Обираємо один з файлів або називаємо його |

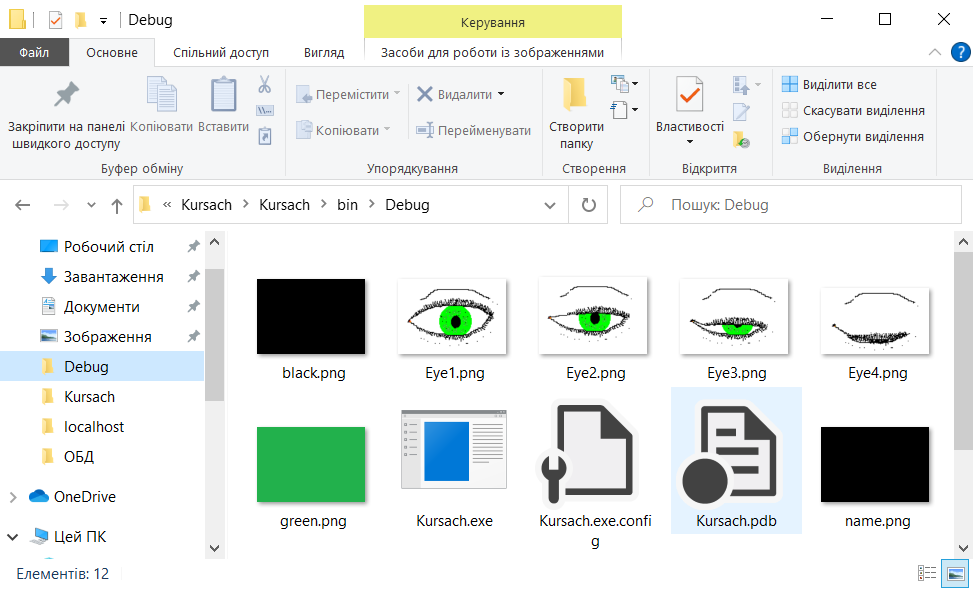


Рис. 3.6 в – Ілюстрація збереженого файлу name.png

|  |  |
| --- | --- |
|  |  |
| Рис. 3.7 а – Обираємо пункт меню «Закрити» | Рис. 3.7 б – Якщо файл не був змінений, то зберігання не відбувається |

У разі зміни відкривається Рис. 3.6 б й далі дочірня форма закриється.

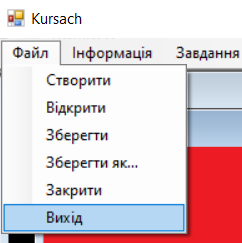


Рис. 3.8 – Обираємо пункт меню «Вихід»

Далі через цикл повторюється перебір усіх форм.

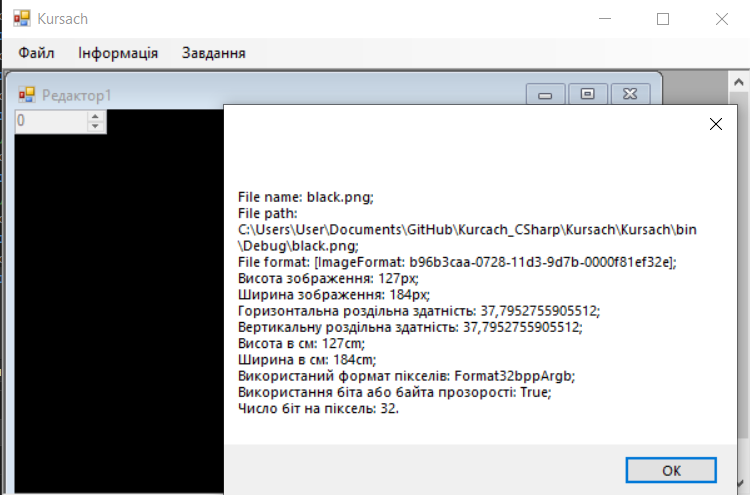


Рис. 3.9 – Обираємо пункт меню «Інформація»



## **3.2 Крок 2. Тестування індивідуального завдання**

Протестуємо завдання 1. Приближення. Ось так воно виглядає:

Додайте в програму інструмент лупа для збільшення і зменшення зображення. Лупа збільшує зображення при натисканні лівою кнопкою миші, лупа зменшує зображення при натисканні правою кнопкою миші. При виборі цього інструменту через меню або панель інструментів повинен змінюватися курсор миші при знаходженні над клієнтської областю вікна. Кнопка на панелі інструментів і пункт меню повинні бути позначені при виборі відповідного інструменту.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Рис. 3.10 а – Обираємо пункт меню «Завдання 1». При IsZoomWorking стала true | Рис. 3.10 б – Змінений курсор | Рис. 3.10 в – Обираємо пункт меню «Завдання 1». При IsZoomWorking стала false |

Спробуємо приблизити зображення.

|  |  |
| --- | --- |
|  |  |
| Рис. 3.11 а – Зображення у повному обсязі | Рис. 3.11 б – Приближенне зображення |

Протестуємо завдання 2. Зміна зображення. Ось так воно виглядає:

Підсумовування двох зображень або константи і зображення. Функція imadd (X, Y, Z) підсумовує кожен елемент масиву X з відповідним елементом масиву Y і повертає суму відповідних елементів в результуючий масив Z. X і Y представляють собою масиви чисел із плаваючою комою однакового розміру і однакового формату представлення даних. Результуючий масив Z має той же розмір і формат представлення даних, що і Y, коли Y скаляр формату double. В іншому випадку розмірність і формат представлення даних результуючого масиву Z збігається з масивом X. Коли X і Y є масиви цілих чисел і елементи результуючого масиву перевищують допустимий діапазон, то вони скорочуються або округлюються. Продемонструйте в програмі застосування цієї функції.

|  |  |
| --- | --- |
|  |  |
| Рис. 3.12 а – Обираємо пункт меню «Завдання 2». При IsNumericChangerEnable стала true | Рис. 3.12 б – Обираємо пункт меню «Завдання 2». При IsNumericChangerEnable стала false |

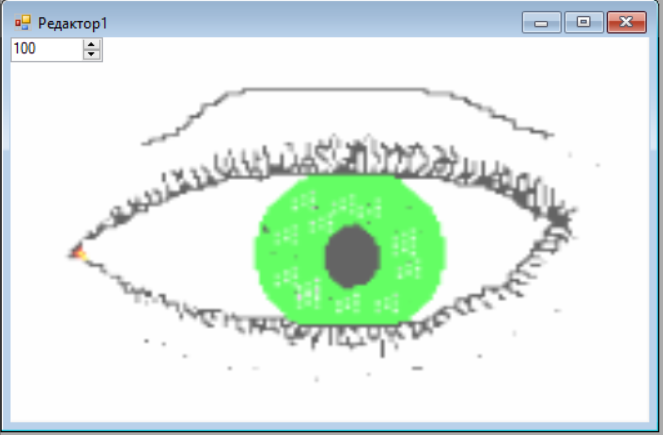


Рис. 3.13 – Змінене зображення з кількістю 100. Зображення, що не було змінене на зображене на Рис. 3.11 а.

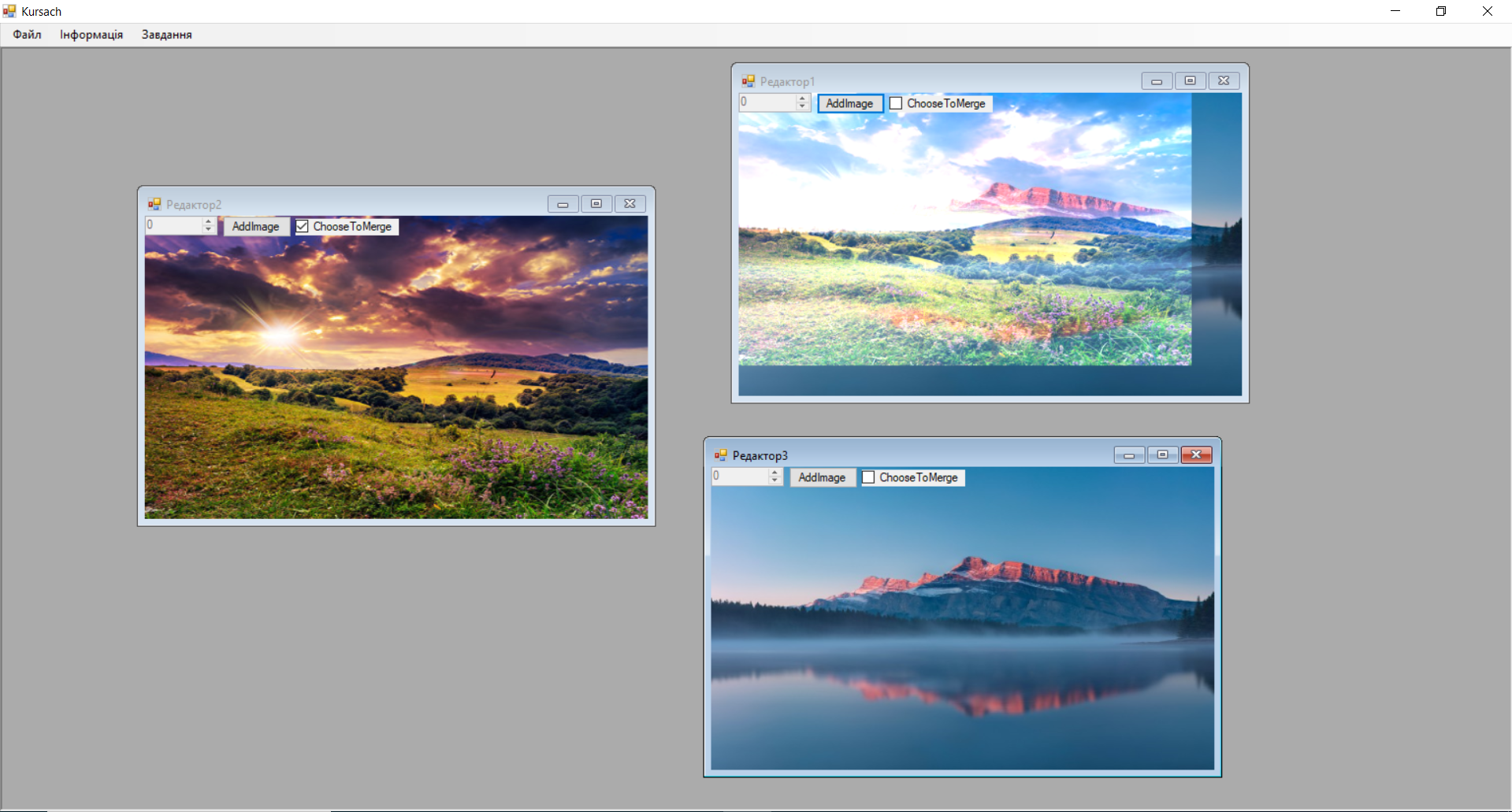


Рис. 3.14 – Об’єднанні зображення у редакторі 1

Усі методи і класи программи працюють корректно.

# **Висновок**

В результаті виконання даної роботи було проведено вивчення принципів побудови системних програм та засвоєно основні засоби та методи конструювання ефективних програм на платформі .NET з використанням методів об'єктно-орієнтованого програмування.

Для розробки програмного забезпечення обробки зображень були використані сучасні технології програмування з використанням різноманітних структур бібліотеки базових класів Framework Class Library (FCL), зокрема класів Graphics, Image та Bitmap. Було проведено дослідження методів та властивостей цих класів, що дозволяють завантаження, збереження і вивід зображень на монітор та принтер, отримання відомості про зображення: формат файлу, розмір, число кольорів, роздільну здатність та багато інших методів обробки зображень.

При створенні додатків було враховано необхідність розробки зручного та зрозумілого для користувача графічного інтерфейсу з використанням різноманітних елементів керування, таких як меню, дочірні форми, користувальницькі та стандартні діалогові вікна, з можливістю задавати їх властивості та оброблювати події. Також було розглянуто методи читання даних з файлів та запис результатів обробки у файли з використанням потоків та класів вводу-виводу.

Отже, під час виконання даної роботи було успішно виконано поставлені завдання, отримано практичні навички програмування на платформі .NET з використанням методів об'єктно-орієнтованого програмування та різноманітних структур бібліотеки базових класів FCL, що дозволяють розробляти ефективні програми об робки зображень зі зручним та зрозумілим графічним інтерфейсом. Також було здійснено дослідження та вивчено принципи побудови системних програм, які можуть бути корисними при розробці різноманітних програмних продуктів.

У ході виконання була розробленна програма по перегляду зображень.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

**1.** Язык программирования [Електронний ресурс]. Режим до-ступу: https://ru.wikipedia.org/wiki/Язык\_программирования — Загол. з екрана.

**2**. Методичні вказівки до виконання курсового проекту з дисципліни “Системне програмування” / Л.Я.Мартинович, І.В. Пономарьов, Ю.М. Рибка – Дніпро: ДНУ, 2017. – 60 с., іл.

**3**. Литвинов О. А., Герасимов В. В., Карпенко Н. В. Об’єктно-орієнтована розробка інформаційних систем. Монографія. / О. А. Литвинов, В. В. Герасимов, Н. В. Карпенко – Д.: Ліра. – 2018. –448 с.

**4**. Programming languages — C. ISO/IEC 9899:2017 [Електрон-ний ресурс]. Режим доступу: https://web.archive.org/web/20181230041359if\_/http://www.open-std.org/jtc1/sc22/wg14/www/abq/c17\_updated\_proposed\_fdis.pdf — Загол. з екрана.

**5**. Методичні вказівки до виконання курсових робіт з дисципліни «Програмування» / Н. В. Карпенко, В. В. Герасимов — Д.: Ліра, 2020. — 52 с.

**6**. Карпенко Н. В., Герасимов В. В. Сучасний підхід до програмування на мові С від нульового до просунутого рівня : навч. посіб. / Н. В. Карпенко, В. В. Герасимов – Д.: Ліра, 2022. – 340 с.

**7.** Троелсен, Эндрю. Язык программирования С# 2010 и платформа .Net 4.0, 5-е издание.: Пер. с англ. - М.: ООО «И. Д. Вильямc», 2011. - 1392 с.

**8.** Шилдт Г. Полное руководство по С# 4.0.: Пер. с англ. - М.: ООО «И. Д. Вильямc», 2012. - 1047 с.

**9.** Іванова Н. О., Мартинович Л. Я. «Створення Windows-форм мовою програмування С#» /Навчально-методичний посібник, В.: Оксамит, Дніпропетровськ, 2007, 144 с.

**10.** Библиотека классов .NET Framework. Работа с растровыми и векторными изображениями с использованием классов Image, Bitmap и Metafile. https://msdn.microsoft.com/ru-ru/library/3ke1f63h(v=vs.110).aspx

# **ДОДАТКИ**

**Додаток А**

*Лістинг «FormGeneral.cs»*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using System.IO;

using System.Drawing.Imaging;

using System.Diagnostics;

using System.Runtime.Remoting.Messaging;

namespace Kursach

{

public partial class FormGeneral : Form

{

// Буфер для картинки

public Bitmap ImageBuffer { get; private set; }

// Буфер картинки для добавления картинок

public Bitmap ImageBufferAdd { get; private set; }

// Путь к картинке

public string ImagePath { get; private set; }

// Активен ли зум

public bool IsZoomWorking { get; private set; }

// Изменитель картинки

public bool IsNumericChangerEnable { get; private set; }

// Количество редакторов

private int \_nextFormNumber { get; set; }

public FormGeneral()

{

InitializeComponent();

// Присваиваем свойствам значения

ImageBufferAdd = null;

ImageBuffer = null;

ImagePath = string.Empty;

\_nextFormNumber = 1;

IsZoomWorking = false;

// Присваиваем название для формы

Text = Const.ProgramName;

// Форма является контейнером для дочерних MDI-форм

IsMdiContainer = true;

// Встановити назву для image файлу

OpenFileDialogWindow.FileName = Const.ImageFileName;

// Встановлюємо фільтр для файлів

OpenFileDialogWindow.Filter = Const.FilesFilter;

}

//1 Point

private void CreateToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Создание нового экземпляра дочерней формы

FormChild newChild = new FormChild(this, "Редактор" + \_nextFormNumber++);

// Вывод созданной формы

newChild.Show();

// Подзагрузка Image

newChild.UploadImageToBuffer();

// Включаем changer картинки

if(IsNumericChangerEnable is true)

newChild.ChangePoint\_NumericUpDown.Enabled = true;

else

newChild.ChangePoint\_NumericUpDown.Enabled = false;

newChild.UploadToBuffer += UploadToBuffer;

}

private void OpenToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Если результат равен Cancel, то выходим

if (OpenFileDialogWindow.ShowDialog() == DialogResult.Cancel)

return;

// Получаем выбранный файл

ImagePath = OpenFileDialogWindow.FileName;

try

{

// Пытаемся открыть файл

using (var bmp = (Bitmap)Image.FromFile(ImagePath))

ImageBuffer = new Bitmap(bmp);

}

catch

{

// Если файл не открылся

MessageBox.Show("Cannot find file " + ImagePath + "!", Text, MessageBoxButtons.OK, MessageBoxIcon.Hand);

return;

}

// Если нет активной формы, спросим про создание

if (ActiveMdiChild is null)

{

DialogResult dialogResult =

MessageBox.Show(Const.Messages.CreateFormToViewImage, Const.Messages.Attention, MessageBoxButtons.YesNo);

// Если пользователь ответил да, то создаем формочку

if (dialogResult == DialogResult.Yes)

CreateToolStripMenuItem\_Click(null, null);

}

else {

// Определение активного дочернего MDI-окна

FormChild activeChildForm = (FormChild)this.ActiveMdiChild;

// Подгружаем картинку в буфер

activeChildForm.UploadImageToBuffer();

// Перерисовываем графику

activeChildForm.Invalidate();

}

// Очищаем буфер

ImageBuffer = null;

// Очищаем путь к картинке

ImagePath = string.Empty;

}

private void SaveToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Проверяем есть ли активная форма

if (ActiveMdiChild is null)

{

MessageBox.Show(Const.Messages.ActiveFormIsNull);

return;

}

// Получаем активную форму

FormChild activeChildForm = (FormChild)this.ActiveMdiChild;

// Получаем ответ от пользователя

DialogResult dialogResult =

MessageBox.Show(Const.Messages.ImageInThisFile, Const.Messages.Attention, MessageBoxButtons.YesNo);

// Если ответ да, то сохраняем

if (dialogResult == DialogResult.Yes)

Save(activeChildForm, activeChildForm.ImagePath);

}

private void SaveAsToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Проверяем есть ли активная форма

if (ActiveMdiChild is null)

{

MessageBox.Show(Const.Messages.ActiveFormIsNull);

return;

}

// Создаем объект SaveFileDialog

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

// Проводим настройку

saveFileDialog1.Filter = Const.FilesFilter;

// Определение активного дочернего MDI-окна

FormChild activeChildForm = (FormChild)this.ActiveMdiChild;

// Проверяем, нажал ли пользователь ОК, и тогда сохраняем

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

Save(activeChildForm, saveFileDialog1.FileName);

}

private void CloseToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Проверяем есть ли активная форма

if (ActiveMdiChild is null)

{

MessageBox.Show(Const.Messages.ActiveFormIsNull);

return;

}

// Определение активного дочернего MDI-окна

FormChild activeChildForm = (FormChild)this.ActiveMdiChild;

// Сохраняем значение, если пользователь того хочет

GetAnswerToSaveAndSave(activeChildForm);

// Закрытие окна

activeChildForm.Close();

}

private void ExitToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Проверяем есть ли активная форма и закрываем приложение, если нет их

if (ActiveMdiChild is null)

Application.Exit();

// Перебираем массив дочерних форм

foreach (FormChild item in MdiChildren)

{

// Выбираем каждую форму

item.Select();

// Сохраняем картинку

GetAnswerToSaveAndSave(item);

}

//Закрываем приложение

Application.Exit();

}

private void GetAnswerToSaveAndSave(FormChild formChild) {

// Если файл был изменен, сохраняем значения

if (formChild.IsChanged is true)

{

// Проверяем, нажал ли пользователь ОК, и тогда сохраняем

DialogResult dialogResult =

MessageBox.Show(Const.Messages.ImageInThisFile, Const.Messages.Attention, MessageBoxButtons.OKCancel);

// Проверяем, нажал ли пользователь ОК, и тогда сохраняем

if (dialogResult == DialogResult.OK)

Save(formChild, formChild.ImagePath);

}

else

MessageBox.Show(Const.Messages.FileNotWasChanged);

}

private void Save(FormChild activeChildForm, string path)

{

// Проверяем пуст ли буфер картинки

if (activeChildForm.ImageBuffer is null)

{

MessageBox.Show(Const.Messages.ImageBufferIsNull);

return;

}

// Проверяем пуст ли путь к картинке

if (string.IsNullOrWhiteSpace(path))

{

MessageBox.Show(Const.Messages.IsNullOrWhiteSpace);

return;

}

// Если в блоке будет ошибка, прокинуть ее через MessageBox

try

{

// Удаляем файл

File.Delete(path);

// Сохраняем файл

activeChildForm.ImageBuffer.Save(path, activeChildForm.ImageBuffer.RawFormat);

//Закрываем форму

activeChildForm.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, ex.GetType().Name);

}

}

//2 Point

private void InformationToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Проверяем есть ли активная форма

if (ActiveMdiChild is null) {

MessageBox.Show(Const.Messages.ActiveFormIsNull);

return;

}

// Получаем активную форму

FormChild activeChildForm = (FormChild)this.ActiveMdiChild;

//Проверяем, существует ли картинка на форме

if (activeChildForm.ImageBuffer is null) {

MessageBox.Show(Const.Messages.ImageBufferIsNull);

return;

}

// Цифра для формулы

double inch = 2.54;

// Переменные dpi

float dpiX, dpiY, dpiBase = 96;

using (var g = new Control().CreateGraphics())

{

dpiX = g.DpiX;

dpiY = g.DpiY;

}

// Создадим переменную текста Информации

string text =

$"\nFile name: {Path.GetFileName(activeChildForm.ImagePath)};" +

$"\nFile path: {activeChildForm.ImagePath};" +

$"\nFile format: {activeChildForm.ImageBuffer.RawFormat};" +

$"\nВисота зображення: {activeChildForm.ImageBuffer.Height}px;" +

$"\nШирина зображення: {activeChildForm.ImageBuffer.Width}px;" +

$"\nГоризонтальна роздільна здатність: {activeChildForm.ImageBuffer.HorizontalResolution / inch};" +

$"\nВертикальну роздільна здатність: {activeChildForm.ImageBuffer.VerticalResolution / inch};" +

$"\nВисота в см: {activeChildForm.ImageBuffer.Height \* dpiX / dpiBase}cm;" +

$"\nШирина в см: {activeChildForm.ImageBuffer.Width \* dpiY / dpiBase}cm;" +

$"\nВикористаний формат пікселів: {activeChildForm.ImageBuffer.PixelFormat};" +

$"\nВикористання біта або байта прозорості: {Image.IsAlphaPixelFormat(activeChildForm.ImageBuffer.PixelFormat)};" +

$"\nЧисло біт на піксель: {Image.GetPixelFormatSize(activeChildForm.ImageBuffer.PixelFormat)}.";

MessageBox.Show(text);

}

//3 Point

private void Task1ToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Состояние зума переключается

IsZoomWorking = !IsZoomWorking;

// Активируем лупу, на новом состоянии

if (IsZoomWorking is true)

{

MessageBox.Show("Режим лупа активовано!");

// Меняем курсор

this.Cursor = Cursors.Hand;

}

// Выключаем лупу, на новом состоянии

else {

MessageBox.Show("Режим лупа деактивовано!");

// Меняем курсор

this.Cursor = Cursors.Default;

}

}

private void Task2ToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Проверяем есть ли активная форма

if (ActiveMdiChild is null)

{

MessageBox.Show(Const.Messages.ActiveFormIsNull);

return;

}

// Состояние Changer переключается

IsNumericChangerEnable = !IsNumericChangerEnable;

// Если Changer активирован

if (IsNumericChangerEnable is true)

{

MessageBox.Show("Changer змін активовано! Змінюйте кольори!");

// Включаем его на всех формах

foreach (FormChild item in MdiChildren)

item.ChangePoint\_NumericUpDown.Enabled = true;

}

// Если Changer деактивирован

else

{

MessageBox.Show("Changer змін деактивовано!");

// Выключаем его на всех формах

foreach (FormChild item in MdiChildren)

item.ChangePoint\_NumericUpDown.Enabled = false;

}

}

private void UploadToBuffer(Bitmap newImage)

{

// Выключаем на всех формах выбранный елемент

foreach (FormChild item in MdiChildren)

// Если елемент включен

if (item.Merged\_CheckBox.Checked is true)

// Если устанавливаемая картинка не равна новой картинке либо буфер пустой

if (item.ImageBuffer != newImage || item.ImageBuffer is null)

item.Merged\_CheckBox.Checked = false;

ImageBufferAdd = newImage;

}

private void FormGeneral\_Load(object sender, EventArgs e){}

}

}

**Додаток Б**

*Лістинг «FormChild.cs»*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Imaging;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kursach

{

public partial class FormChild : Form

{

// Буфер картинки

public Bitmap ImageBuffer { get; private set; }

// Событие подгрузки картинки в буфер

public Action<Bitmap> UploadToBuffer { get; set; }

// Путь к картинке

public string ImagePath { get; private set; }

// Статус измененной картинки

public bool IsChanged { get; private set; }

// Статус приближения картинки

public bool IsZoomed { get; private set; }

// Статус добавленного изображения

public bool IsAddImage { get; private set; }

// Коеффициент приближения

private int \_coefficient { get; set; }

public FormChild(FormGeneral parent, string caption)

{

InitializeComponent();

// Присваиваем публичным свойствам значения

ImageBuffer = null;

ImagePath = string.Empty;

IsChanged = false;

IsZoomed = false;

//По дефолту выключаем ChangePoint\_NumericUpDown

ChangePoint\_NumericUpDown.Enabled = false;

// Присваиваем приватным свойствам значения

\_coefficient = 2;

// Подписываем события

this.Paint += FormChild\_Paint;

this.Resize += FormChild\_Resize;

// Присваивание контейнеру родителя данной формы

this.MdiParent = parent;

// Задание заголовка

this.Text = caption;

}

private void FormChild\_Load(object sender, EventArgs e) { }

public void FormChild\_Paint(object sender, PaintEventArgs e)

{

// Проверяем буфер на имение картинки

if (ImageBuffer is null) {

MessageBox.Show(Const.Messages.ImageBufferIsNull);

return;

}

// Переменная графики

Graphics graphics = e.Graphics;

// Переменная сохраняющая картинку

Bitmap userImage = null;

// Если картинка была изменена

if (IsChanged is true || IsAddImage is true)

// Получаем измененную картинку

userImage = GetChangedImage();

else // Получаем не измененную картинку

userImage = ImageBuffer;

//if(((FormGeneral)this.MdiParent).)

// Отслеживаем статус картинки

if (IsZoomed is false)

// Прорисовываем картинку

graphics.DrawImage(userImage, 0, 0, ClientRectangle.Width, ClientRectangle.Height);

else

// Прорисовывем приближенную картинку

graphics.DrawImage(userImage, -PointToClient(Cursor.Position).X, -PointToClient(Cursor.Position).Y,

ClientRectangle.Width \* \_coefficient, ClientRectangle.Height \* \_coefficient);

// Освобождаем ресурсы графики

graphics.Dispose();

// Если изменения присутствуют

if (IsChanged is true)

{

// Получаем ответ от пользователя

DialogResult dialogResult =

MessageBox.Show(Const.Messages.DownloadImageToBuffer, Const.Messages.Attention, MessageBoxButtons.YesNo);

// Подгружаем новую картинку в буфер

if(dialogResult == DialogResult.Yes)

ImageBuffer = userImage;

}

}

public Bitmap GetChangedImage()

{

Bitmap newImage = null;

if (IsAddImage is true) {

// Выключаем переключатель

IsAddImage = false;

// Подтаскиваем изображение

newImage = AddImageByPixels();

}

else

newImage = AddPixelsToImage();

return newImage;

}

private Bitmap AddImageByPixels() {

// Создаем переменную, что хранит изображение родителя

Bitmap imageFromParent = ((FormGeneral)this.MdiParent).ImageBufferAdd;

Bitmap bitmap = ImageBuffer;

Color color;

for (int x = 0; x < bitmap.Width; x++)

for (int y = 0; y < bitmap.Height; y++)

{

if (imageFromParent.Height > y && imageFromParent.Width > x)

// Создаем новый цвет для пикселя

color = Color.FromArgb(

GetNumPixel(bitmap.GetPixel(x, y).R, imageFromParent.GetPixel(x, y).R),

GetNumPixel(bitmap.GetPixel(x, y).G, imageFromParent.GetPixel(x, y).G),

GetNumPixel(bitmap.GetPixel(x, y).B, imageFromParent.GetPixel(x, y).B)

);

else color = Color.FromArgb(

GetNumPixel(bitmap.GetPixel(x, y).R, 0),

GetNumPixel(bitmap.GetPixel(x, y).G, 0),

GetNumPixel(bitmap.GetPixel(x, y).B, 0)

);

// Устанавливаем новый пиксель

bitmap.SetPixel(x, y, color);

}

return bitmap;

}

private Bitmap AddPixelsToImage()

{

// Создаем временную картинку для изменений

Bitmap bitmap = ImageBuffer;

for (int x = 0; x < bitmap.Width; x++)

for (int y = 0; y < bitmap.Height; y++)

{

// Создаем новый цвет для пикселя

Color color = Color.FromArgb(

GetNumPixel(bitmap.GetPixel(x, y).R),

GetNumPixel(bitmap.GetPixel(x, y).G),

GetNumPixel(bitmap.GetPixel(x, y).B)

);

// Устанавливаем новый пиксель

bitmap.SetPixel(x, y, color);

}

return bitmap;

}

private byte GetNumPixel(byte num)

{

byte newNum = 0;

try

{

newNum = Convert.ToByte(num + ChangePoint\_NumericUpDown.Value);

}

catch (OverflowException)

{

newNum = 255;

}

return newNum;

}

private byte GetNumPixel(byte num, byte addNum)

{

byte newNum = 0;

try

{

newNum = Convert.ToByte(num + addNum);

}

catch (OverflowException)

{

newNum = 255;

}

return newNum;

}

public void UploadImageToBuffer()

{

// Проверяем буфер картинки

if (((FormGeneral)this.MdiParent).ImageBuffer is null) {

return;

}

// Проверяем путь картинки

if (((FormGeneral)this.MdiParent).ImagePath == string.Empty) {

MessageBox.Show(Const.Messages.IsNullOrWhiteSpace);

return;

}

// Присваиваем в поля данные

ImageBuffer = ((FormGeneral)this.MdiParent).ImageBuffer;

ImagePath = ((FormGeneral)this.MdiParent).ImagePath;

}

private void FormChild\_Resize(object sender, EventArgs e) {

Invalidate();//перерисовать

}

private void FormChild\_Click(object sender, EventArgs e)

{

// Если приближение не активировано выходим

if(((FormGeneral)this.MdiParent).IsZoomWorking is false)

return;

// Если приближение в этой форме не было произведено, отмечаем

if (IsZoomed is false)

IsZoomed = true;

else

// Если приближение в этой форме было произведено, выключаем

IsZoomed = false;

Invalidate();//перерисовать

}

private void ChangePoint\_NumericUpDown\_ValueChanged(object sender, EventArgs e)

{

// Отмечаем изменение картинки

if (IsChanged is false)

IsChanged = true;

Invalidate();//перерисовать

}

private void Merged\_CheckBox\_CheckedChanged(object sender, EventArgs e)

{

// Проверяем ImageBuffer пустой ли он

if (ImageBuffer is null) {

MessageBox.Show(Const.Messages.ImageBufferIsNull);

return;

}

if (Merged\_CheckBox.Enabled is true)

UploadToBuffer.Invoke(this.ImageBuffer);

else

UploadToBuffer.Invoke(null);

}

private void AddImage\_Button\_Click(object sender, EventArgs e)

{

// Создаем переменную, что хранит изображение родителя

Bitmap imageFromParent = ((FormGeneral)this.MdiParent).ImageBufferAdd;

// Проверяем ImageBuffer пустой ли он

if (ImageBuffer is null)

{

MessageBox.Show(Const.Messages.ImageBufferIsNull);

return;

}

// Проверяем ImageBufferAdd пустой ли он

if (imageFromParent is null)

{

MessageBox.Show(Const.Messages.ImageBufferAddIsNull);

return;

}

// Проверяем, одинаковы ли картинки

if (imageFromParent == ImageBuffer)

{

MessageBox.Show(Const.Messages.ImagesIsEqual);

return;

}

IsAddImage = true;

Invalidate();

}

}

}

**Додаток В**

*Лістинг «Const.cs»*

using System.Runtime.Serialization;

namespace Kursach

{

public class Const {

public const string FilesFilter =

"Image files(\*.bmp;\*.jpg;\*.png;\*.gif;\*.tiff)|\*.bmp;\*.jpg;\*.png;\*.gif;\*.tiff";

public const string ImageFileName = "image.png";

public const string ProgramName = "Kursach";

public partial class Messages {

public const string Attention = "Увага!";

public const string CreateFormToViewImage = "Ви бажаєте створити форму для відображення картинки?";

public const string ImageInThisFile = "Ви хочете зберегти картинку в тому ж файлі або ви не хочете зберігати зовсім?";

public const string DownloadImageToBuffer = "Ви хочете підвантажити змінену версію цієї картинки в ImageBuffer?";

public const string FileNotWasChanged = "Файл не змінено.";

public const string ActiveFormIsNull = "Активної форми немає!";

public const string ImageBufferIsNull = "Буфер картинки порожній! Можливо, її не існує на формі!";

public const string ImageBufferAddIsNull = "Буфер картинки на головній формі, що повинна об`єднуватися з цією формою пустий!";

public const string ImagesIsEqual = "Не можна об'єднувати однакові зображення!";

public const string IsNullOrWhiteSpace = "Не задано шлях для збереження!";

}

}

}