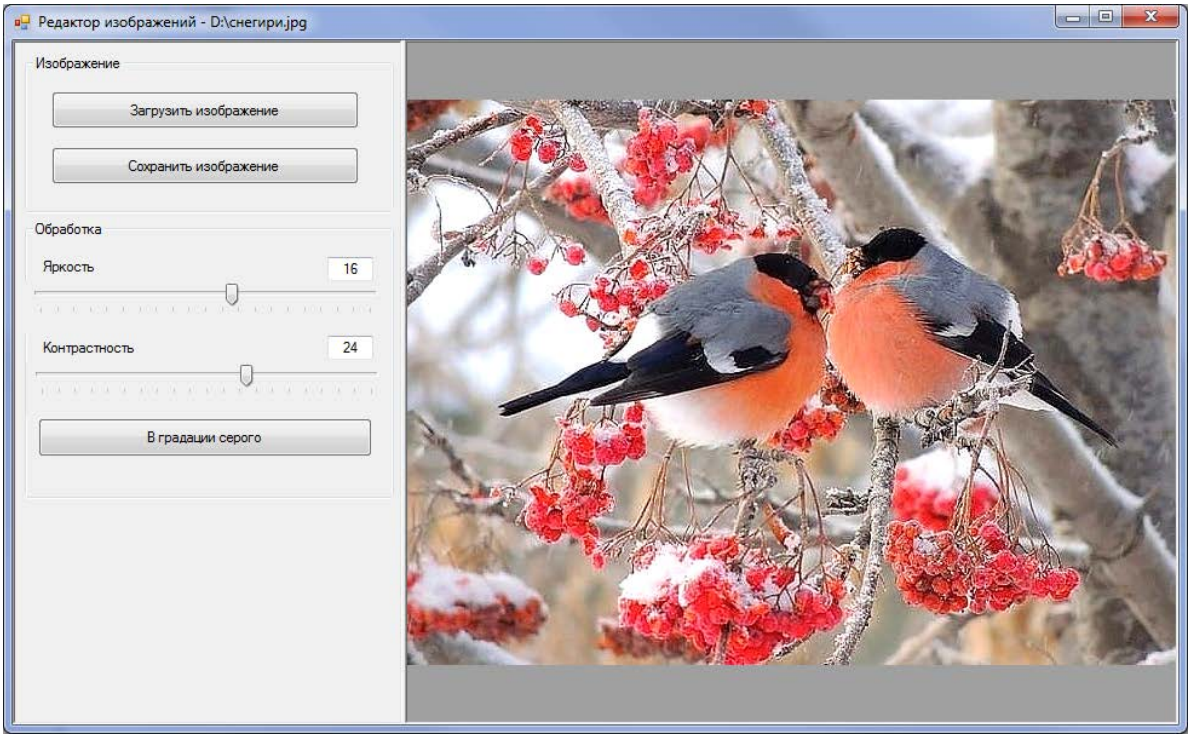
Лабораторная работа №1

«Создание программы изменения цветовых оттенков изображения на языке C#»

Задание

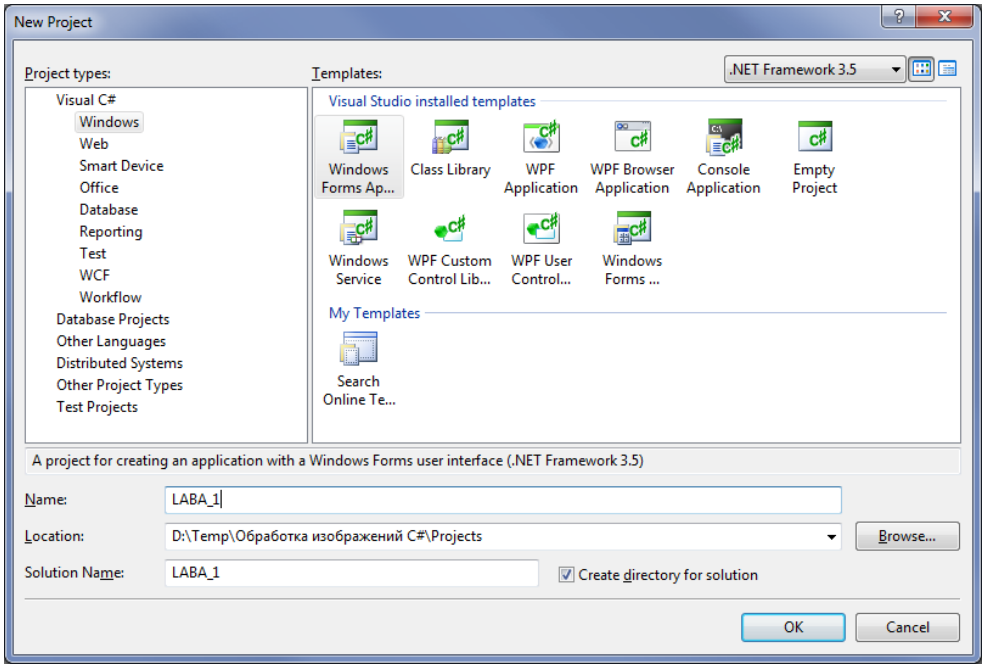
Создать программу обработки изображения, загружаемого из файла, с целью изменения его цветовых оттенков.

Ожидаемый результат



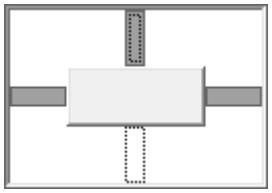
Создание проекта

- Запустить Visual Studio.
- Выбрать пункт меню по созданию нового проекта.
- Создать проект на C# - Windows Forms Application.



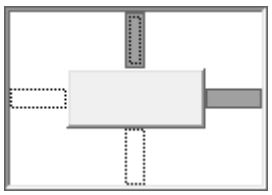
Настройка главного окна программы

- В свойствах формы для свойства Text указать «Редактор изображений»
- Настроить свойство, чтобы форма при своём появлении при запуске программы выводилась в центре экрана.
- Бросить на форму из панели Toolbox - элемент SplitterContainer. Задача этого элемента в том, чтобы разбить форму на 2 области (размер которых можно будет менять). В одной области будет располагаться панель управления свойствами редактируемого изображения, в другой – визуализироваться само изображение.
- Для свойства Dock элемента SplitterContainer установить значение Fill.
- Для сплиттера для свойства BorderStyle указать значение Fixed3D.
- Для панели сплиттера Panel2 можно в качестве фона указать какой-нибудь темный цвет.
- В панель Panel1 сплиттера добавить из панели элементов компонент GroupBox. В данном элементе будут располагаться кнопки управления загрузкой файла изображения для редактирования, а также кнопка сохранения в файл финального, отредактированного результата.
- Позиционировать элемент в панели подходящим образом.
- Для его свойства Text указать значение «Изображение».
- Для свойства Anchor установить значение, как показано на следующем рисунке. Данное свойство позволяет «привязывать» границы элемента управления к границам области, в которой он располагается. Так, если установить данное свойство, как показано на рисунке, то при изменении положения сплиттера (влево/вправо) элемент «Изображение» будет тоже растягиваться/стягиваться, оставаясь всегда на одном и том же расстоянии по бокам от границ.

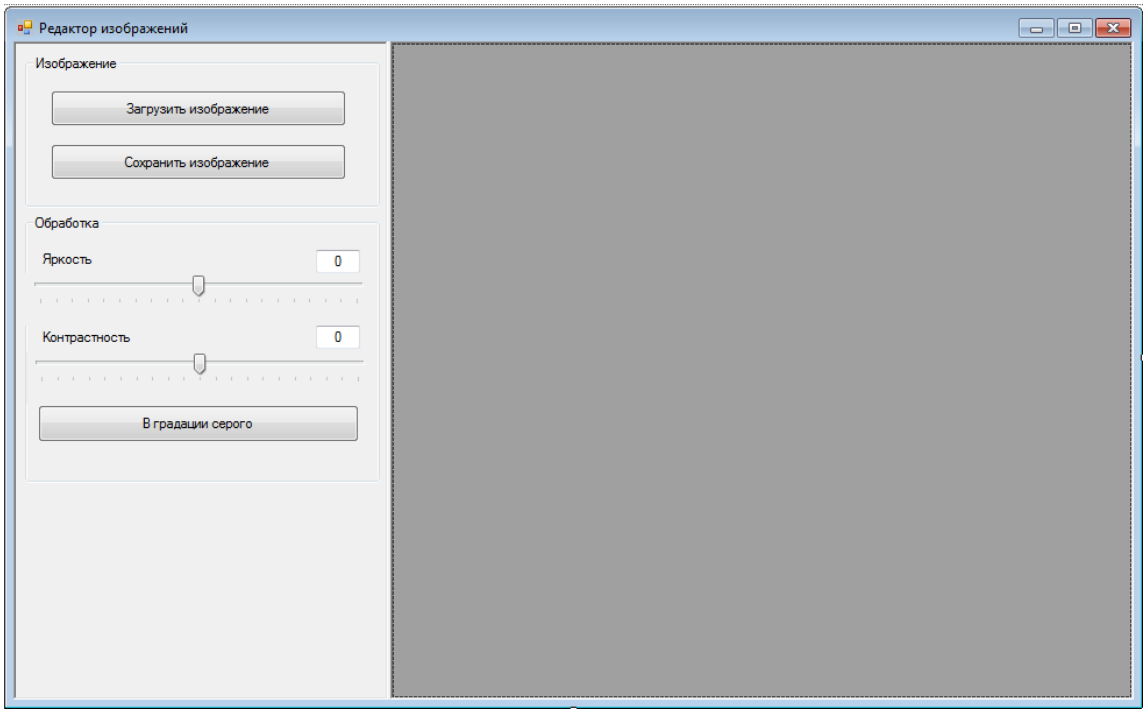


- Из панели компонентов добавить в элемент «Изображение» пару кнопок.
- Расположить их подходящим образом.
- Первой дать имя «butLoad» и название «Загрузить изображение».
- Второй дать имя «butSave» и название «Сохранить изображение».
- Для обеих кнопок установить их свойство Anchor как и для элемента «Изображение».
- В панель Panel1 сплиттера, чуть ниже добавить из панели элементов компонент GroupBox. В данном элементе будут располагаться элементы управления свойствами изображения.
- Позиционировать элемент в панели подходящим образом.
- Для его свойства Text указать значение «Обработка».
- Для его свойства Anchor установить значение, как и для элемента «Изображение».
- В область элемента «Обработка» из панели добавить пару контролов TrackBar. Данные элементы будут управлять яркостью и контрастностью редактируемого изображения. Дать им соответственно имена: trackBarBright и trackBarContrast.
- Для свойств Anchor этих трекбаров установить значения, как и для элемента «Изображение».
- Для элемента trackBarBright установить свойства минимального значения -100, максимального 100, текущего (Value) - 0. Для свойства LargeChange установить значение 1.
- Для элемента trackBarContrast соответствующие свойства установить аналогично.
- Из панели элементов добавить пару элементов Label, в которые вписать текст «Яркость» и «Контрастность» соответственно. Расположить их так, чтобы они подписывали соответствующие трекбары.

- Из панели элементов добавить пару элементов TextBox, которые расположить справа и чуть выше трекбаров. Их задача – отображать текущие значения яркости и контрастности редактируемого изображения. Задать их имена соответственно textBoxBright и textBoxContrast.
- Установить для них свойство отображаемого текста по центру элемента, в качестве отображаемого текста – указать 0.
- Для свойства Anchor установить значение, которое бы позволяло «прибить» этот элемент к правой стороне панели «Обработка».



- Чуть ниже трекбаров добавить кнопку, указав в качестве её имени butToGray, а отображаемого текста – «В градации серого». Эта кнопка будет отвечать за преобразование изображения к формату «в оттенках серого».
- Для свойства Anchor этой кнопки установить значение, как и для кнопки «Сохранить изображение».
- В правую панель сплиттера поместить элемент PictureBox. Данный элемент будет визуализировать редактируемое изображение.
- Для свойства Dock этого элемента установить значение Fill. Это позволить полностью вписать пикчебокс в рамки панели.
- Для свойства SizeMode элемента PictureBox установить значение CenterImage. Это позволит отображаемой картинке выводиться всегда по центру элемента управления.
- Скомпилировать приложение и запустить его.
- Вид окна приложения должен быть примерно таким, как показано на следующем рисунке.



Программирование логики приложения

- Определить в классе формы 2 объекта, которые будут хранить ссылки на редактируемое изображение.

```
Bitmap m_bmpMain;     // исходное изображение
Bitmap m_bmpSource;   // копия исходного изображения
```

Загрузка изображения

- Дважды кликнуть по кнопке «Загрузить изображение».
- Создать объект окна диалога, позволяющего выбрать на дисках файл для последующего открытия. Для этого добавить в программу следующую строку.

```
OpenFileDialog openFileDialog = new OpenFileDialog();
```

- Настроить объект, определив его свойства

```
openFileDialog.Filter = "Графические файлы (*.BMP; *.JPG; *.GIF; *.PNG) |
                                        *.BMP;*.JPG;*.GIF;*.PNG";
```

- Открыть диалог, проверить условие выбора файла и нажатия кнопки «Открыть».

```
if (openFileDialog.ShowDialog() == DialogResult.OK)
{

}
```

- В скобки условия вписать следующий код.

```
m_bmpMain = new Bitmap(openFileDialog.FileName);
pictureBox1.Image = m_bmpMain;
```

- Здесь уже физически создается битовое изображение по информации из открываемого файла, которое затем отображается в пикчебоксе в правой части окна.
- Чтобы в заголовке окна отображалась информация о загруженном на редактирование файле, можно вписать следующую строку, которая выводит в заголовок окна имя файла.

```
Text = "Редактор изображений - " + openFileDialog.FileName;
```

Сохранение отредактированного изображения

- Дважды кликнуть по кнопке «Сохранить изображение».
- Создать объект окна диалога, позволяющего выбрать (или создать) файл для сохранения. Для этого добавить в программу следующую строку.

```
SaveFileDialog saveFileDialog = new SaveFileDialog();
```

- Настроить объект, определив его свойства

```
saveFileDialog.Filter = "Графические файлы (*.BMP; *.JPG; *.GIF; *.PNG) |
                                        *.BMP;*.JPG;*.GIF;*.PNG";
```

- Открыть диалог, проверить условие выбора файла и нажатия кнопки «Сохранить».

```
if (saveFileDialog.ShowDialog() == DialogResult.OK)
{

}
```

- В скобки условия вписать следующий код.

```
m_bmpSource.Save(saveFileDialog.FileName);
MessageBox.Show("Изображение сохранено", "Информация", MessageBoxButtons.OK,
                                                MessageBoxIcon.Information);
```

Здесь для битмапа, хранящего результат редактирования, вызывается метод его сохранения с указанием имени файла, которое было получено из диалогового окна. Затем на экран выводится сообщение об успешном сохранении файла.

Функция ToByte

- Создать в классе формы закрытую сервисную функцию ToByte. Задача этой функции – преобразовывать некорректные значения цветов компонентов (RGB) в диапазон 0 – 255. Некорректные значения цветов могут получаться из-за преобразований, связанных с изменением яркости изображения.
- Функция ToByte должна возвращать результат типа int и принимать параметр с именем Val типа int.
- Тело функции должно включать следующие строки кода.

```
if (Val > 255)
    Val = 255;
else
    if (Val < 0) Val = 0;
```

- Функция ToByte в качестве результата должна возвращать значение переменной Val.

Функция GetComponent

- Создать в классе формы закрытую сервисную функцию GetComponent. Задача этой функции изменить цвет текущего пикселя так, чтобы в конечном итоге изменилась контрастность редактируемого изображения.

- Функция GetComponent должна возвращать результат типа int и принимать параметры component типа int и value типа int. Первый параметр – это значение компоненты цвета пикселя (RGB), второй параметр – значение величины контраста, получаемое из трекбара «Контрастность».

- Тело функции должно включать следующие строки. В теле функции реализуется алгоритм изменения контрастности пикселя. Результат выполнения функции – новое значение компонента цвета пикселя.

```
double Pixel;
//Вычисляем общее значение контраста
double Contrast = (100.0 + value) / 100.0;

Pixel = component / 255.0;
Pixel = Pixel - 0.5;
Pixel = Pixel * Contrast;
Pixel = Pixel + 0.5;
Pixel = Pixel * 255;
if (Pixel < 0) Pixel = 0;
if (Pixel > 255) Pixel = 255;

return Convert.ToByte(Pixel);
```

Функция Processing

- Создать в классе формы закрытую сервисную функцию Processing. Задача этой функции – преобразовывать изображение по яркости и контрастности в соответствии с их значениями, задаваемыми с помощью трекбаров.
- Функция Processing ничего не должна возвращать. Принимаемые параметры – BrVal типа int и ConVal типа int.
- Для изменения яркости и контрастности необходимо попиксельно перебрать всё изображение и поменять цвета компонентов его пикселей. Поэтому добавить в функцию цикл прохода по пикселям изображения.

```
for (int x = 0; x < m_bmpSource.Width; ++x)
          for (int y = 0; y < m_bmpSource.Height; ++y)
```

- Чтобы изменить яркость пикселя, нужно к каждому компоненту его цвета добавить некоторую величину. Тогда яркость увеличится. Если отнять величину, яркость уменьшится. Эта величина берется от трекбара «Яркость». При этом может возникнуть ситуация, когда в результате арифметических преобразований цвет компоненты пикселя выйдет из диапазона 0 – 255. В этом случае нужно произветси его коррекцию. Поэтому каждый новый вычисленный цвет нужно передать в функцию ToByte, которая его скорректирует в случае ошибки.
- В тело цикла добавить следующие строки.

```
int r, g, b;

Color curr = m_bmpSource.GetPixel(x, y); // получить цвет пикселя

r = ToByte(curr.R + BrVal);  // задать новое значение компоненты R цвета пикселя
g = ToByte(curr.G + BrVal);  // задать новое значение компоненты G цвета пикселя
b = ToByte(curr.B + BrVal);  // задать новое значение компоненты B цвета пикселя

// сформировать новый цвет из ранее вычисленных компонент
Color next = Color.FromArgb(GetComponent(r, ConVal),
                            GetComponent(g, ConVal),
                            GetComponent(b, ConVal));

m_bmpSource.SetPixel(x, y, next); // установить новый цвет пикселю
```

- Здесь для каждого пикселя считывается его цвет, раскладывается на составляющие RGB, далее с помощью функции ToByte изменяется яркость каждой компоненты данного пикселя.

- Далее формируется цвет этого пикселя, для которого при помощи функции GetComponent изменяется контрастность каждой компоненты пикселя.

- Затем для текущего пикселя редактируемого изображения изменяется его прежний цвет на новый, с уже измененной яркостью и контрастностью (функция SetPixel).

- После циклической обработки всех пикселей изображения можно показать результат, скопировав отредактированное изображение в PictureBox в окне.

```
pictureBox1.Image = m_bmpSource;
```

Функция OnNeedProcessing

- Создать обработчик OnNeedProcessing события ValueChanged для трекбаров «Яркость» и «Контрастность».

- В тело функции-обработчика добавить строку, в которой берется исходное изображение для редактирования и делается его копия. Именно с этой копией и проделываются все процедуры изменения яркости и контрастности. А исходное изображения всегда хранится неизменным.

```
m_bmpSource = (Bitmap)m_bmpMain.Clone();
```

- В тело функции-обработчика добавить строки, которые считывают значения трекбаров «Яркость» и «Контрастность» и выводят эту информацию в соответствующие справочные текстовые элементы.

```
textBoxBright.Text = trackBarBright.Value.ToString();
textBoxContrast.Text = trackBarContrast.Value.ToString();
```

- Далее необходимо вызвать функцию изменения яркости и контрастности изображения в соответствии с их установленными значениями с помощью трекбаров.

```
Processing(trackBarBright.Value, trackBarContrast.Value);
```

Функция перевода изображения в градации серого

- Преобразование к оттенкам серого заключается в получении яркости каждой точки по формуле Y:=0.3*R+0.59*G+0.11*B, а также в последующем копировании полученного значения в каждый из каналов RGB.
- Два раза кликнуть на кнопке «В градации серого».
- С помощью условного оператора и функции MessageBox сделать проверку, действительно ли пользователь хочет изменить свойства изображения.

```
if (MessageBox.Show("Преобразовать изображение к формату в оттенках серого?",
                    "Вопрос",MessageBoxButtons.YesNo,
                             MessageBoxIcon.Question) == DialogResult.Yes)
```

- В случае утвердительного ответа организовать цикл попиксельного перебора изображения (как это тбыло сделано в функции Processing), только вместо битмапа m_bmpSource здесь поставить битмап m_bmpMain. Это делается так потому, что данное преобразование необратимо. Утратив цвета, изображение больше нельзя будет преобразовать в цветную форму. И далее все операции по изменению яркости, контрастности будут уже применяться к преобразованному в градации серого изображению. Поэтому необходимо перевести в серость – именно исходное, базовое изображение.

- В тело цикла добавить строки преобразования.

```
int grey;
Color curr = m_bmpMain.GetPixel(x, y);

grey = (Byte)(0.3 * curr.R + 0.59 * curr.G + 0.11 * curr.B);

Color next = Color.FromArgb(grey, grey, grey);

m_bmpMain.SetPixel(x, y, next);
```

- Поскольку до преобразования возможно изменялась яркость или контрастность изображения, то её необходимо учесть и здесь. Поэтому нужно вызвать функцию применения текущих параметров яркости и контрастности к уже серому изображению.

```
OnNeedProcessing(null, null);
```

- Далее можно заблокировать кнопку вызова преобразования в оттенки серого, поскольку сделав преобразование один раз, следующие разы проделывать его – бесмысленно.

```
butToGray.Enabled = false;
```

Тюнинг программы

- При запуске программы, пока изображение еще не загружено не имеет смысла нажимать кнопки сохранения, преобразования в оттенки серого, изменение положения позунков… Это приведет к ошибкам. Ведь изображение еще не загружено. Поэтому имеет смысл при запуске программы сразу выставить свойство Enabled в значение false для элементов butSave, butToGray, trackBarBright и trackBarContrast.

- И наоборот. Когда изображение загружено, нужно эти элементы теперь разблокировать.

- После запуска программы запустить диспетчер задач Windows.
- Выбрать вкладку «Процессы».
- В таблице найти разрабатываемую программу.
- Определить объем памяти, выделенной системой для программы.

- Несколько раз изменить яркость и контрастность изображения.
- Наблюдать за количеством памяти.
- Видно, что при каждой операции её объем растёт. Это происходит из-за того, что в функции OnNeedProcessing в строке

```
m_bmpSource = (Bitmap)m_bmpMain.Clone();
```

создается клон исходного изображения в памяти, ссылка на него копируется в m_bmpSource. И так происходит всякий раз при изменении яркости или контрастности. Старое изображение-клон нам уже не нужно и мы создаем новое. А старой при этом хранится внутри. И мы еще перезаписываем на него ссылкку новым значением. Получается утечка памяти. Чтобы этого не было, нужно перед созданием копии исходного изображения – удалять старую копию. Это можно сделать, добавив в начале функции OnNeedProcessing строку.

```
if (m_bmpSource != null) m_bmpSource.Dispose();
```

- Промониторить с помощью диспетчера объем памяти теперь.