Преобразование цветного изображения в черно-белое C#

В статье рассмотрим алгоритм перевода цветного изображения в черно-белое и напишем его реализацию на языке C#. Приводится исходный код программы, написанной в Visual Studio – приложение Windows Forms.

Сразу оговоримся, что под преобразованием в черно-белое изображение будем понимать преобразование в оттенки серого. В рамках данного материала эти два понятия используются как синонимы.

Алгоритм преобразования в черно-белое изображение

Каждый пиксель изображения формируется при помощи сочетания четырех каналов: ARGB (Alpha, Red, Green, Blue), альфа-канала, красного, зеленого и синего.

Альфа-канал отвечает за прозрачность пикселя (100% – пиксель полностью непрозрачный, 0% – полностью прозрачный).

Сочетание значений RGB каналов определяет цвет пикселя.

Каждый канал несёт в себе 8 бит информации (1 байт), соответственно интенсивность канала может меняться в диапазоне от 0 до 255. Полностью пиксель занимает в памяти 32 бита (4 байта).

Для того, чтобы **преобразовать цветное изображение в черно-белое**, нужно найти среднее арифметическое значение R, G и B каналов пикселя и затем это значение присвоить RGB каналам этого же пикселя (то есть оно будет одинаковое). Альфа-канал оставляем без изменений.

Такую операцию нужно проделать с каждым пикселем изображения.

Программа для преобразования цветного изображения в черно-белое

Приступим к написанию приложения Windows Forms в среде разработки Visual Studio на языке программирования C#.

Создадим новый проект программы и разместим на форме 3 кнопки (Button) и два контейнера под изображения PictureBox. Кнопки понадобятся для: открытия изображения, преобразования в черно-белое (в оттенки серого) и сохранения преобразованной картинки на компьютер.

Свойства кнопок Text изменим на:

* Открыть
* Ч/Б
* Сохранить

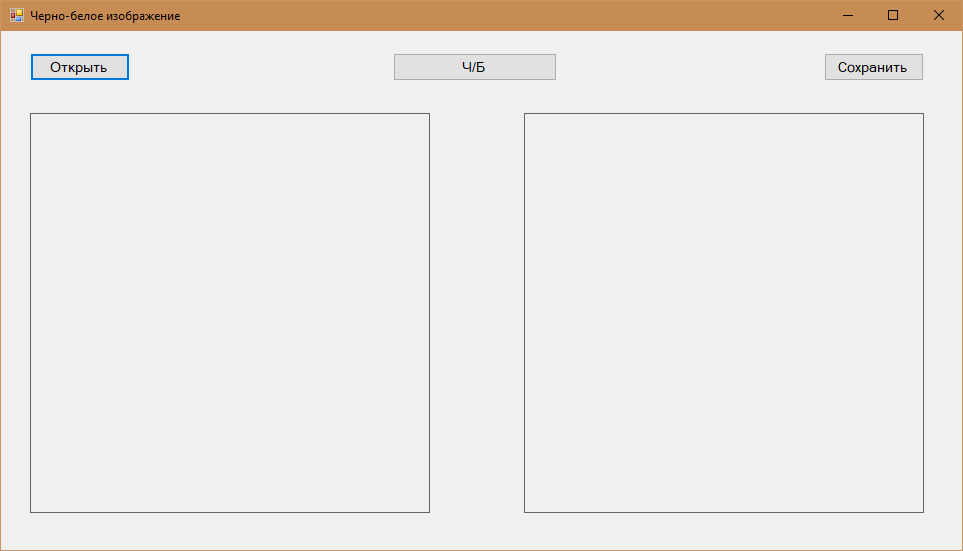
Имена кнопок (Name), используемые при разработке, изменим на:

* openButton
* grayButton
* saveButton

Размеры обоих PictureBox (Size) сделаем равными 400 x 400 пикселей.

Чтобы вокруг границ PictureBox отображалась рамка в виде сплошной тонкой линии, установим значение свойства BorderStyle в положение FixedSingle.

Также у PictureBox свойство SizeMode сделаем равным Zoom. Данная настройка позволит автоматически масштабировать отображаемые изображения в соответствии с размером PictureBox, при этом сохраняя соотношения сторон исходной картинки.

[](https://vscode.ru/wp-content/uploads/2017/07/Interfeys-programmyi.png)

Перейдём к написанию кода. Сначала закодируем обработку нажатий кнопок “Открыть” и “Сохранить”:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | // кнопка Открыть  private void openButton\_Click(object sender, EventArgs e)  {      // диалог для выбора файла      OpenFileDialog ofd = new OpenFileDialog();      // фильтр форматов файлов      ofd.Filter = "Image Files(\*.BMP;\*.JPG;\*.GIF;\*.PNG)|\*.BMP;\*.JPG;\*.GIF;\*.PNG|All files (\*.\*)|\*.\*";      // если в диалоге была нажата кнопка ОК      if (ofd.ShowDialog() == DialogResult.OK)      {          try          {              // загружаем изображение              pictureBox1.Image = new Bitmap(ofd.FileName);          }          catch // в случае ошибки выводим MessageBox          {              MessageBox.Show("Невозможно открыть выбранный файл", "Ошибка",                  MessageBoxButtons.OK, MessageBoxIcon.Error);          }      }  }    // кнопка Сохранить  private void saveButton\_Click(object sender, EventArgs e)  {      if (pictureBox2.Image != null) // если изображение в pictureBox2 имеется      {          SaveFileDialog sfd = new SaveFileDialog();          sfd.Title = "Сохранить картинку как...";          sfd.OverwritePrompt = true; // показывать ли "Перезаписать файл" если пользователь указывает имя файла, который уже существует          sfd.CheckPathExists = true; // отображает ли диалоговое окно предупреждение, если пользователь указывает путь, который не существует          // фильтр форматов файлов          sfd.Filter = "Image Files(\*.BMP)|\*.BMP|Image Files(\*.JPG)|\*.JPG|Image Files(\*.GIF)|\*.GIF|Image Files(\*.PNG)|\*.PNG|All files (\*.\*)|\*.\*";          sfd.ShowHelp = true; // отображается ли кнопка Справка в диалоговом окне          // если в диалоге была нажата кнопка ОК          if (sfd.ShowDialog() == DialogResult.OK)          {              try              {                  // сохраняем изображение                  pictureBox2.Image.Save(sfd.FileName);              }              catch // в случае ошибки выводим MessageBox              {                  MessageBox.Show("Невозможно сохранить изображение", "Ошибка",                      MessageBoxButtons.OK, MessageBoxIcon.Error);              }          }      }  } |

Теперь напишем код кнопки, преобразующей цветное изображение в оттенки серого:

C#

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | // кнопка Ч/Б  private void grayButton\_Click(object sender, EventArgs e)  {      if (pictureBox1.Image != null) // если изображение в pictureBox1 имеется      {          // создаём Bitmap из изображения, находящегося в pictureBox1          Bitmap input = new Bitmap(pictureBox1.Image);          // создаём Bitmap для черно-белого изображения          Bitmap output = new Bitmap(input.Width, input.Height);          // перебираем в циклах все пиксели исходного изображения          for (int j = 0; j < input.Height; j++)              for (int i = 0; i < input.Width; i++)              {                  // получаем (i, j) пиксель                  UInt32 pixel = (UInt32)(input.GetPixel(i, j).ToArgb());                  // получаем компоненты цветов пикселя                  float R = (float)((pixel & 0x00FF0000) >> 16); // красный                  float G = (float)((pixel & 0x0000FF00) >> 8); // зеленый                  float B = (float)(pixel & 0x000000FF); // синий                  // делаем цвет черно-белым (оттенки серого) - находим среднее арифметическое                  R = G = B = (R + G + B) / 3.0f;                  // собираем новый пиксель по частям (по каналам)                  UInt32 newPixel = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G << 8) | ((UInt32)B);                  // добавляем его в Bitmap нового изображения                  output.SetPixel(i, j, Color.FromArgb((int)newPixel));              }          // выводим черно-белый Bitmap в pictureBox2          pictureBox2.Image = output;      }  } |

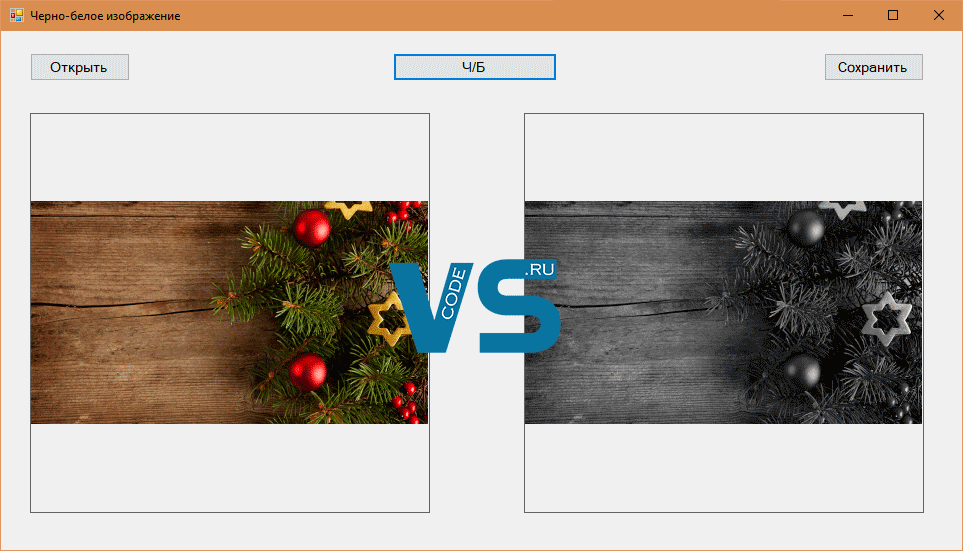
Отдельно следует сказать про хранение значения пикселя в программе и про получение значений отдельных каналов.

Пиксель мы храним в структуре UInt32. Она предназначена для хранения беззнакового целого 4-х байтового (32 бита) числа. Диапазон значений такого числа от 0 до 232 – 1; такой диапазон идеально подходит для хранения значения пикселя – он туда помещается полностью, поскольку занимает также 32 бита.

Для получения значения конкретного канала используется побитовое умножение значения пикселя на битовую маску для соответствующего цвета с последующим сдвигом вправо на нужное количество бит (строки 17-19).

Сборка пикселя из каналов воедино происходит с использованием операций сдвига влево (для каждого канала) и последующим логическим сложением всех компонент (строка 23). Альфа-канал примем равным 0xFF000000 (FF16 = 25510).

Проверим работу программы:

[](https://vscode.ru/wp-content/uploads/2017/07/Preobrazovanie-v-cherno-beloe.png)