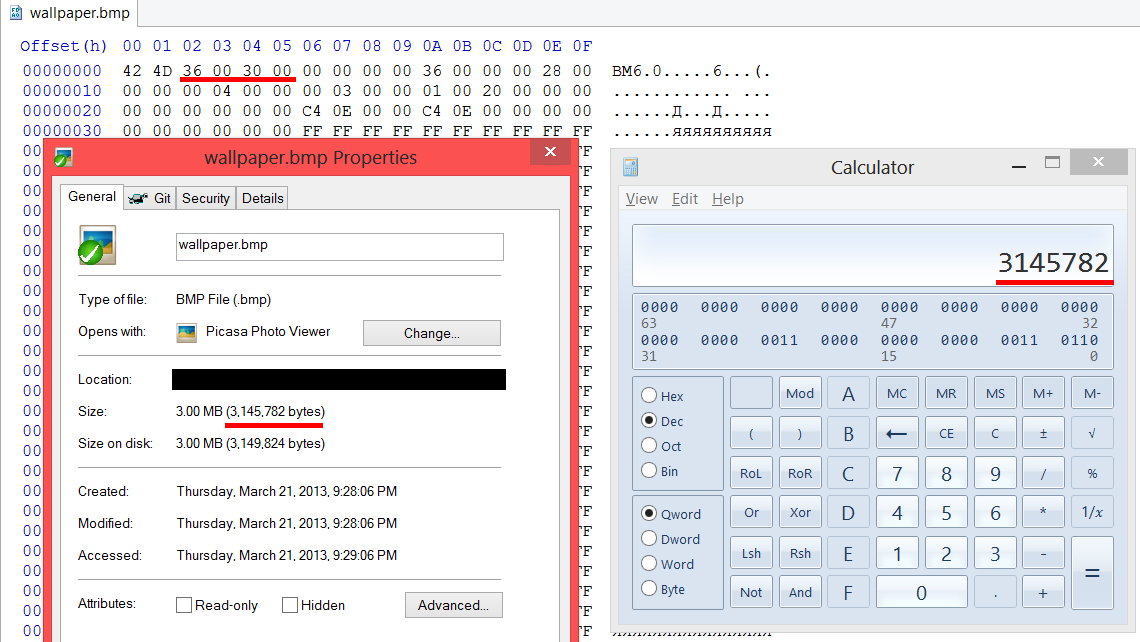
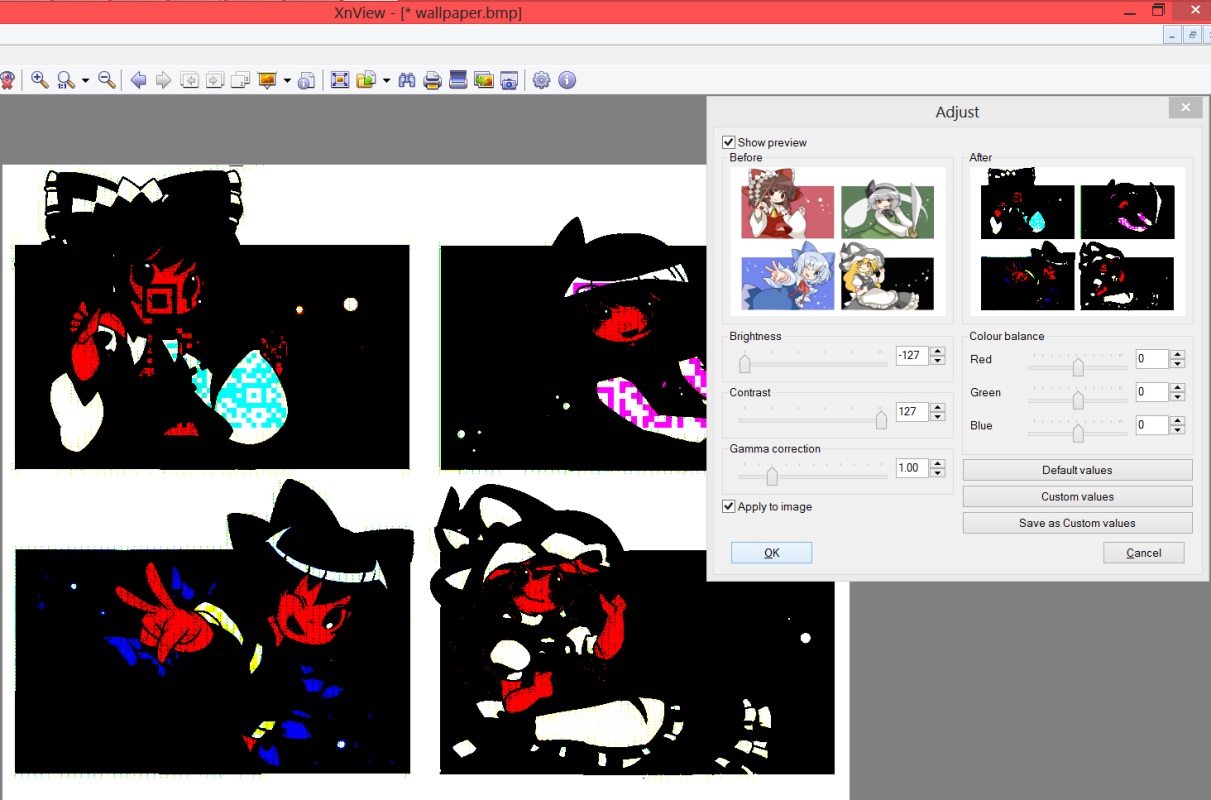
Разбор задачи E

Для начала заметим, что формат картинки – bmp, поэтому возможности по записыванию метаданных в картинку весьма ограниченны и их можно довольно быстро проверить: после картинки в файле ничего не записано, в заголовках только рабочая информация. Значит, если тут скрыта какая-то информация, она находится в самом изображении.



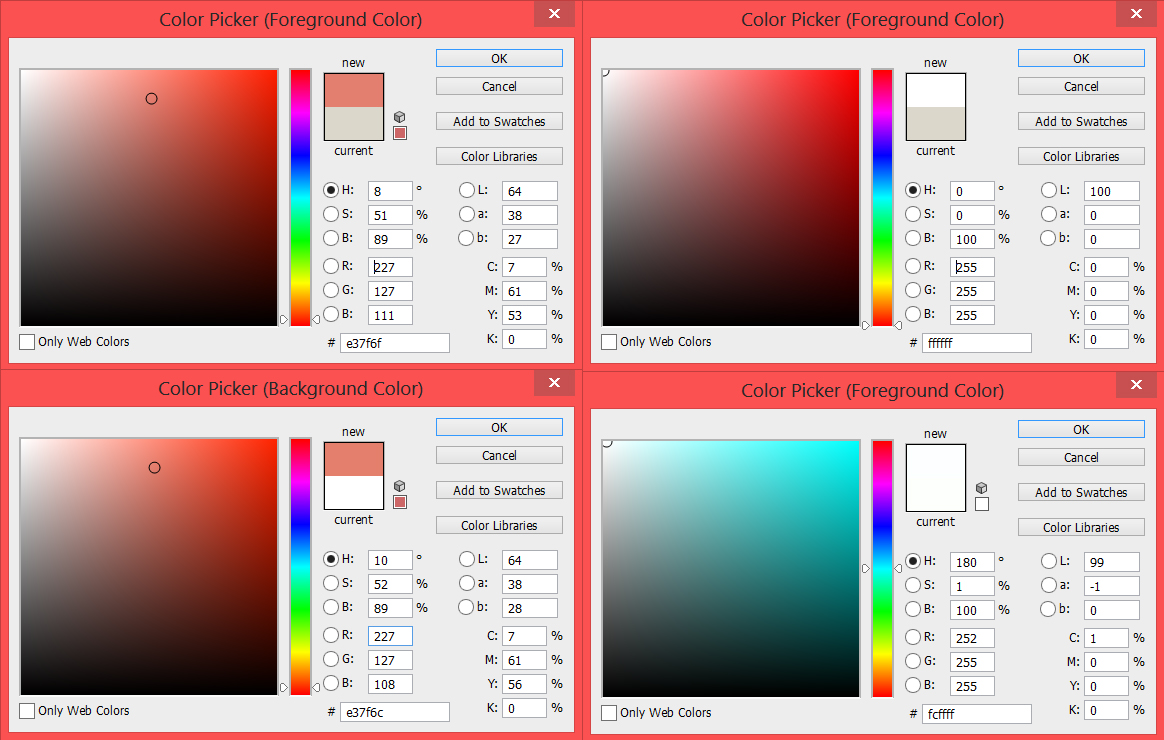
На картинке показано, что bmp считает, что в нем записано 0x300036 байт, и действительно, размер такой, ничего не было добавлено.

Каким образом можно скрыть информацию в изображении? В bmp – только в самих пикселях, то есть в информации о цвете. Но осмысленная информация в бинарном формате, записанная в пиксели, плохо сочетается с эстетичностью данной нам картинки, значит, возможно, мы просто ее не различаем из-за несовершенства нашего зрения. Попробуем изменить параметры картинки так, чтобы малейшая разница между соседними пикселями была крайне заметна. Это можно сделать, в любом графическом редакторе, например, в бесплатном XnView. Увеличим до максимума контрастность, уменьшим до минимума яркость.



Заметим, что у нас появились кусочки чего-то, подозрительно напоминающих QR-коды. Базовая идея следующего шага очевидна – нужно достать эти QR-коды и посмотреть, что в них записано. Но варианты, как это делать, очень разнятся.

Можно, воспользовавшись тем, что пиксели с QR-кодом немного, но отличаются от обычных пикселей, постепенно отделить их от остальных инструментами наподобие «заливки» или «волшебной палочки». Но этот способ неэффективен и оставляем его желающим.

Лучше для начала взглянуть на цвета как на наборы из чисел. Для начала заметим, что изображение 32 битное, а значит, что цвет состоит из четырех компонент – красного, зеленого, синего и прозрачности (альфа-канал). Рассмотрим цвета тех пикселей, которые точно в QR-коде и тех, которые не в QR-коде.

Сверху указаны пиксели не из QR-кода, снизу, соответственно, из QR-кода. Заметим, что у пикселей, которые находятся вне QR-кода, все цвета нечетные, а у пикселей в QR-коде одна из компонент четна. Причем левые два пикселя взяты с синего изображения(с красного бантика), а правые два пикселя – с красного. Мало того, если посмотрим на битовую запись компонент, то можно заметить, что не просто последний бит (отвечающий за четность), но два последних равны нулю, если пиксель в QR-коде, и единицам, если пиксель за пределами. Таким образом, напишем простую программу, которая зануляет нечетные компоненты (написанная ниже программа написана на C#). Таким образом, получаем картинку с четкими QR-кодами, который нетрудно прочитать своим смартфоном (или найти в интернете сервис для распознавания QR-кодов).

