1. Запросить у пользователя в консоли количество оттенков красного, зеленого и синего цвета, к которым будут приводиться изображения.
2. Выделить память под массивы images, notpreparedimages, V.
3. images – матрица, которая будет хранить эталонные изображения, переведенные в бинарную комбинацию.

,

m – количество эталонных изображений;

n – количество элементов бинарной комбинации;   
n = <количество пикселей в изображении> \* <количество оттенков, введенных пользователем> \* 3.

1. notpreparedimages – 4-мерный массив, который будет хранит. эталонные изображения с количеством оттенков цветов, введенные ранее. Данный массив необходим для вывода эталонных изображений на экран монитора.

Первая индекс массива - количество эталонных изображений;

Второй и третий индекс – координаты пикселя данного изображения;

Четвертый индекс – цвет пикселя («1» - красный, «2» - зеленый, «3» - синий);

Значение массива – интенсивность данного цвета данного пикселя данного изображения.

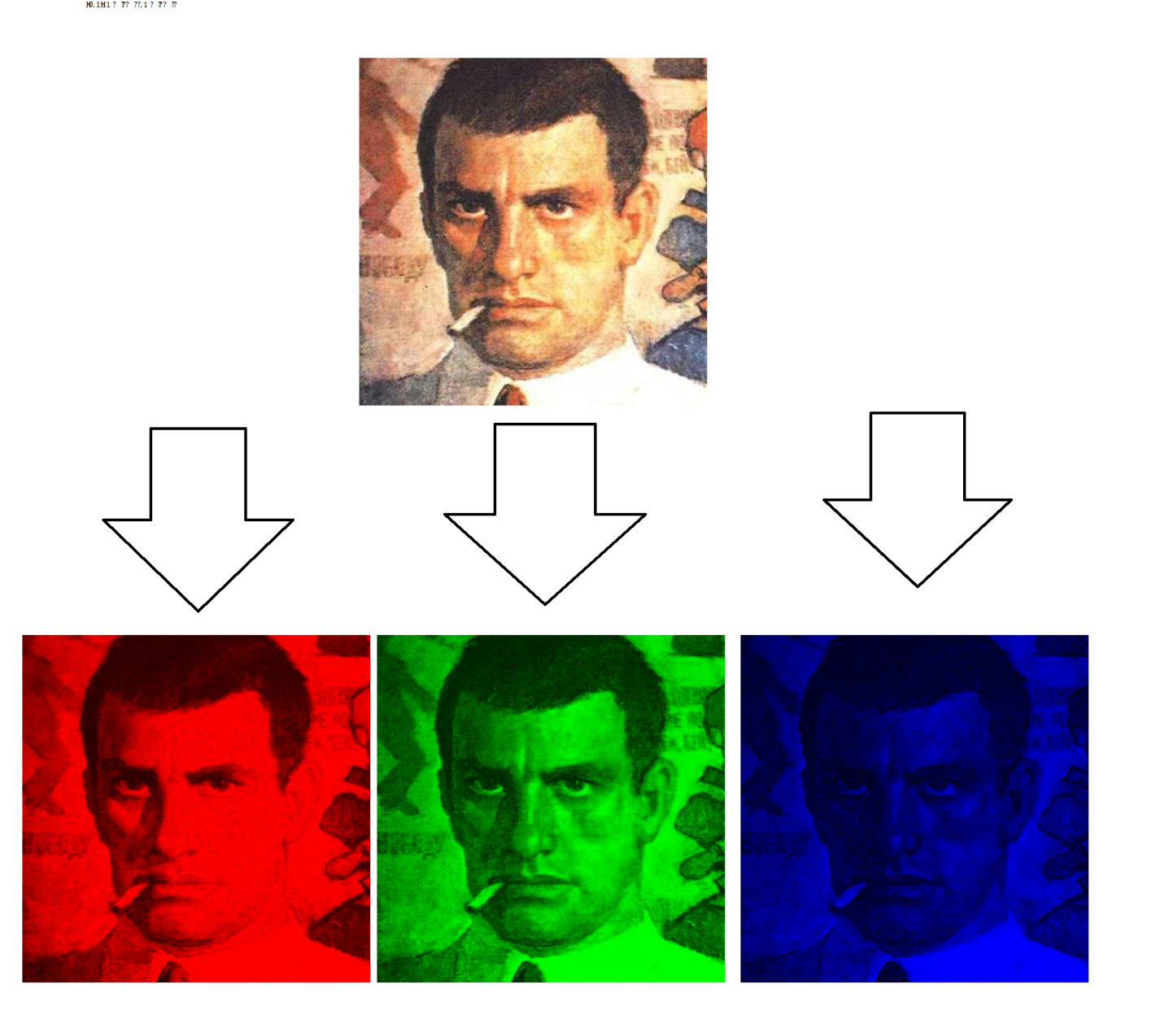
1. V - матрица, необходимая для хранения одного из изображений.

Первый столбец хранит значения интенсивностей красного цвета каждого пикселя изображения, второй – значения интенсивностей зеленого цвета, третий – синего.

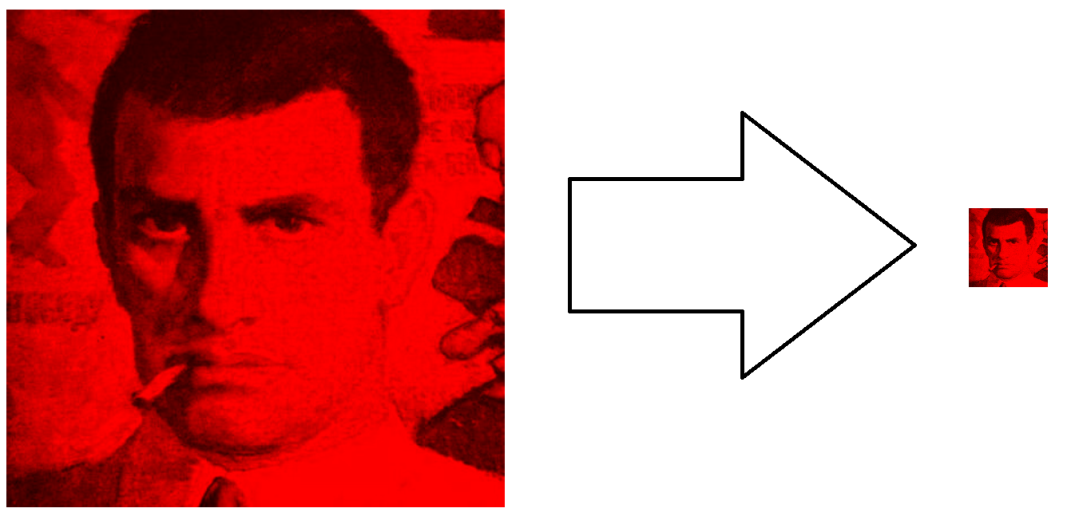
**Работа с эталонным изображением.**

Производится необходимая работа с каждым эталонным изображением для его приведения к бинарной комбинации.

1. Читаем из jpg-файла изображение и помещаем в три различные матрицы, где каждая матрица хранит значения интенсивностей пикселей определенного цвета.

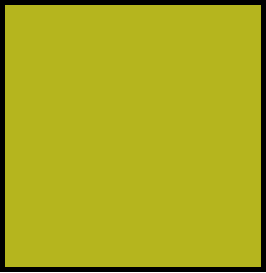


1. **Приведение эталонных изображений к стандартной форме размеру 100х100.**

Приводим матрицы к размеру 100\*100 с помощью метода ближайшего соседа.

1. Приводим матрицы к вектор-строкам длиной 10000.
2. Приводим каждую вектор-строку к бинарной вектор-строке, где каждому элементу массива (пикселю) ставится в соответствие своя бинарная комбинация.

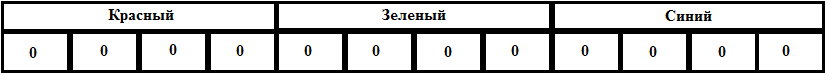
Рассмотрим метод создания бинарной последовательности на примере одного пикселя:



Интенсивность красного цвета – 181, зеленого -181, синего – 30.

Пусть предлагается разбить каждый из цветов на 4 градации.

Для данного пикселя создается бинарный вектор из 12 элементов.



Вначале все значения массива являются нулевыми.

Затем изменяем значение на одном из «красных» индексов (от 1 до 4) на единицу по следующей формуле.

где Index – индекс бинарного вектора, где значение меняется на единицу;  
Red – интенсивность красного цвета у пикселя.

Промежуточный итог виден на следующей таблице:



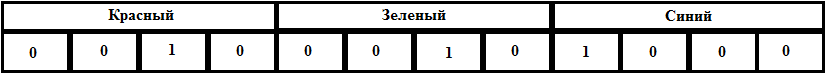
Аналогичным образом изменяем значения на одном из «зеленых» (от 5 до 8) и на одном из «синих» (от 9 до 12) индексов на единицу по следующим формулам:

Green - интенсивность зеленого цвета у пикселя;

Blue - интенсивность синего цвета у пикселя;

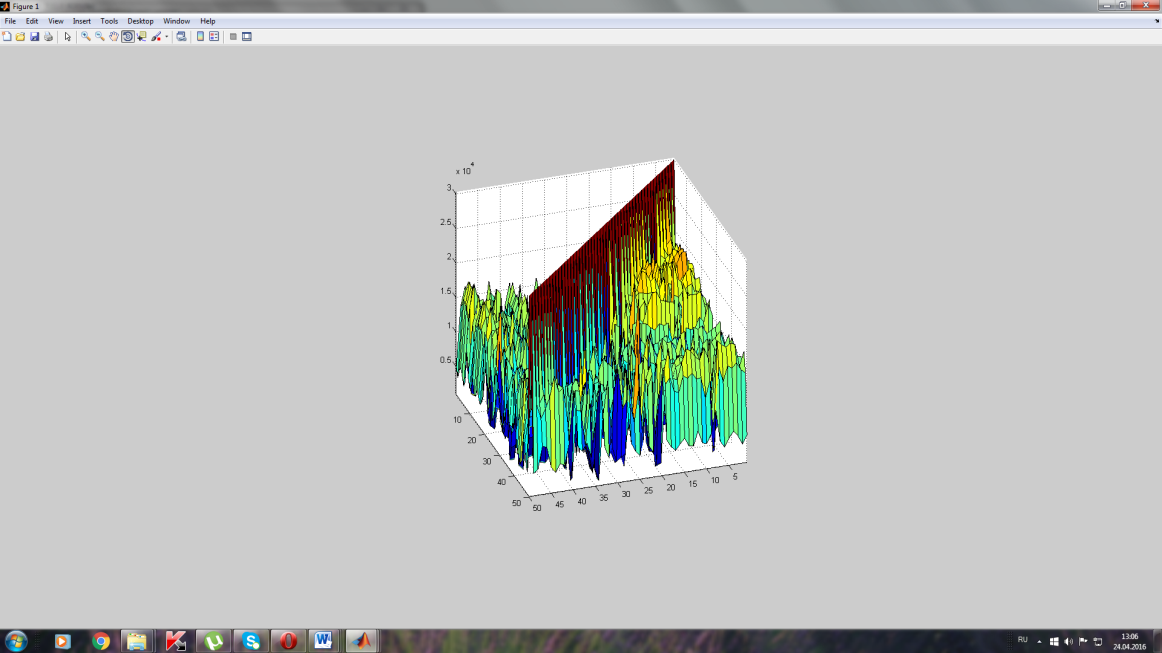
**Окончательный результат.**

Итого получается следующий вектор-бинарная строка или код.



На этом обработка эталонных изображений заканчивается.

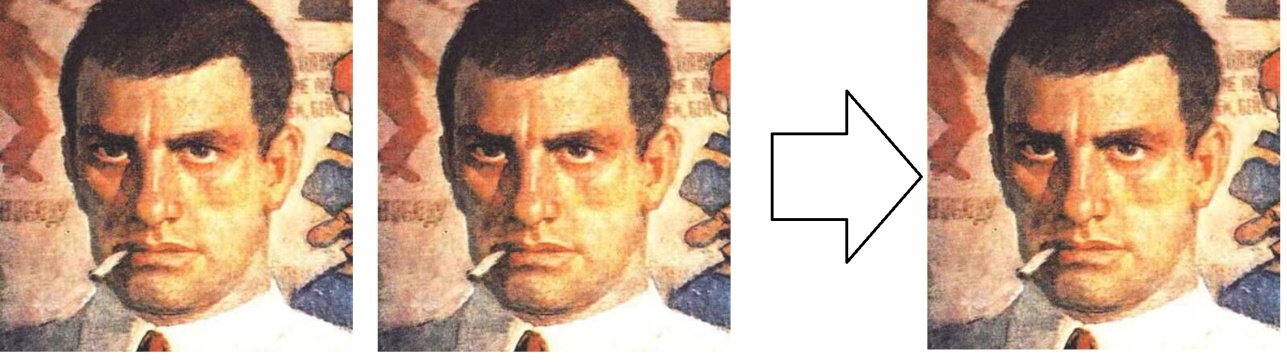
Преимуществом данного способа получения бинарной строки является возможность получения нормированной диагонали матрицы A \* AT. На диагонали должно получаться 30 000.



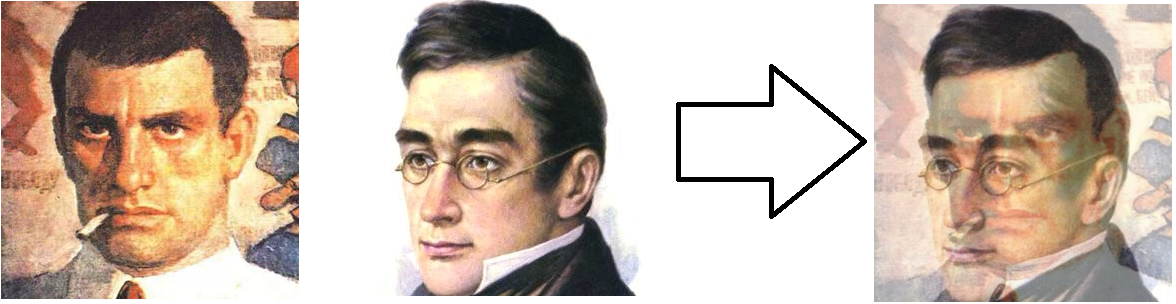
**Работа с тестовым изображением.**

1. Запрашиваем у пользователя посредством диалогового окна тестовое изображение формата jpg для его распознавания.
2. Производится необходимая работа с тестовым изображением для его приведения к бинарной комбинации (аналогичное пункту 3).
3. На основе избирательного нейрона производится сравнение тестового изображения со всеми эталонными. На основе данных сравнений находится самое похожее эталонное изображение, и происходит его вывод на экран с кратким описанием, либо выводится в консоли сообщение о неудачном распознании (если порог распознания не превышен).

Пример сравнения тестового изображения с наиболее похожим на него эталонным:



Пример сравнения тестового изображения с непохожим на него эталонным:



Приложение 1.

**Метод ближайшего соседа.**

Метод ближайшего соседа заключается в следующем. Исходное изображение имеет большое количество пикселей (большее 10000). Изображение подготовлено в виде квадратной формы. Распознавание расчитано на исходные объекты 100х100=10000 пикселей. Поэтому необходимо приведение исходного изображения к стандартной форме. Это преобразование осуществляется с помощью метода ближайшего соседа. В Matlab 7 метод ближайшего соседа реализуется с помощью функции imresize.

Идея метода ближайшего соседа заключается в следующем:

Группа близлежащих пикселей заменяется одним пикселем, наиболее близким к этой группе. Близость оценивается с помощью критерия близости.

Один из типовых подходов заключается в разбиении данного отображения прямоугольной сеткой на квадраты, содержащие больше 1 пикселя. Например, сетку 100х100=10000 пкиселей разбиваем на прямоугольники 70х70=4900 пикселей. При этом в каждый прямоугольник попадает от 1 до 3 пикселей. Простейшим способом является взятие среднего арифметического интенсивностей пикселей, попавших внутрь элемента более крупной сетки. Другие способы описаны в программах реализации данного метода.

Для каждого пикселя конечного изображения выбирается один пиксель исходного, наиболее близкий к его положению с учетом масштабирования.