Задание к работе.

Разработать программу, моделирующую алгоритм кластеризации k-средних. Исходными данными является изображение, содержащее объекты разного цвета на однотонном фоне. В программе должны быть реализованы следующие процедуры с выводом результата на экран:

– загрузка изображения;

– задание вручную числа кластеров k (число кластеров определяется преподавателем);

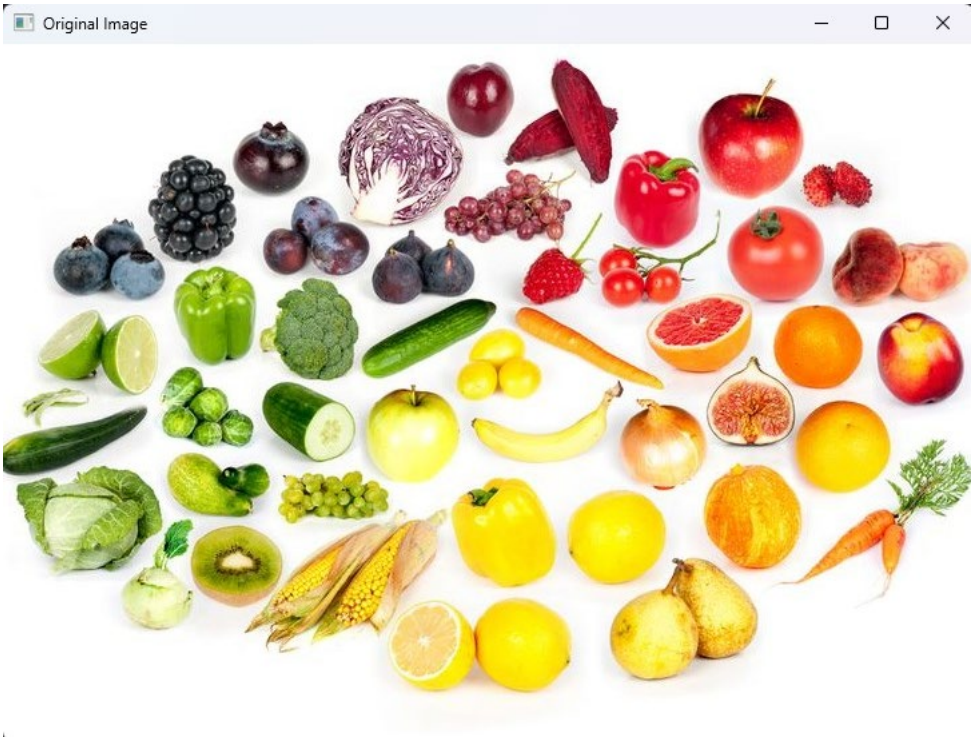
– реализация алгоритма k-средних, где в качестве признаков используются значения R, G, B пикселей; результатом является изображение-маска, содержащая k цветов, соответствующее сформированным кластерам.

Выполнение работы.

1. Загрузка изображения image, с помощью метода cv2.imread().

```
11    # загрузка изображения
12    image = cv2.imread("D:/cv_projects/lab2/fruits_vegs.jpg")
13
14    show_res("Original Image", image)
```

Результат отображения:



2. Задание количества кластеров, на которые будет разбито изображение. Значение преобразуется в целое число с помощью int().

```
16  # задаем количество кластеров
17  k = int(input("Введите количество кластеров (k): "))
```

3. Изменение формы изображения к двумерному массиву пикселей, где каждая строка соответствует пикселю, а каждое значение в строке - это цветовой компонент (RGB). Используется reshape((-1, 3)), где -1 позволяет автоматически рассчитать нужное количество строк. После этого массив преобразуется в тип float32, что потребуется для работы алгоритма k-средних.

```
19  # приведение формы изображения к двумерному массиву пикселей
20  pixel_values = image.reshape((-1, 3))
21  pixel_values = np.float32(pixel_values)  # преобразование во float
```

4. Определение критериев завершения алгоритма k-средних, состоящие из двух условий: максимальное количество итераций и минимальное изменение.

```
23  # определение критериев завершения и выполнение алгоритма k-средних
24  criteria_end = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)
25  _, labels, centers = cv2.kmeans(
26      pixel_values, k, None, criteria_end, 10, cv2.KMEANS_RANDOM_CENTERS
27  )
```

Затем вызывается метод kmeans, который выполняет кластеризацию пикселей изображения и возвращает результаты: labels (метки, указывающие, к какому кластеру принадлежит каждый пиксель), centers (координаты центров кластеров после завершения алгоритма).

Передаваемые значения: pixel_values — массив, представляющий значения пикселей изображения, которые будут кластеризованы;

k — количество кластеров, на которые нужно разбить данные;

None — параметр, который указывает, что начальные центры кластеров будут случайно инициализированы;

criteria_end — ранее созданные критерии завершения, которые определяют, когда алгоритм должен остановиться;

10 — это количество повторений алгоритма, чтобы избежать попадания в локальные минимумы; алгоритм будет выполняться 10 раз, и результатом будет лучший из них.

cv2.KMEANS_RANDOM_CENTERS — метод инициализации центров кластеров случайным образом.

5. Центры кластеров преобразуются обратно в тип uint8, необходимый для правильного отображения изображения.

```
29    # преобразование центров обратно в uint8
30    centers = np.uint8(centers)
```

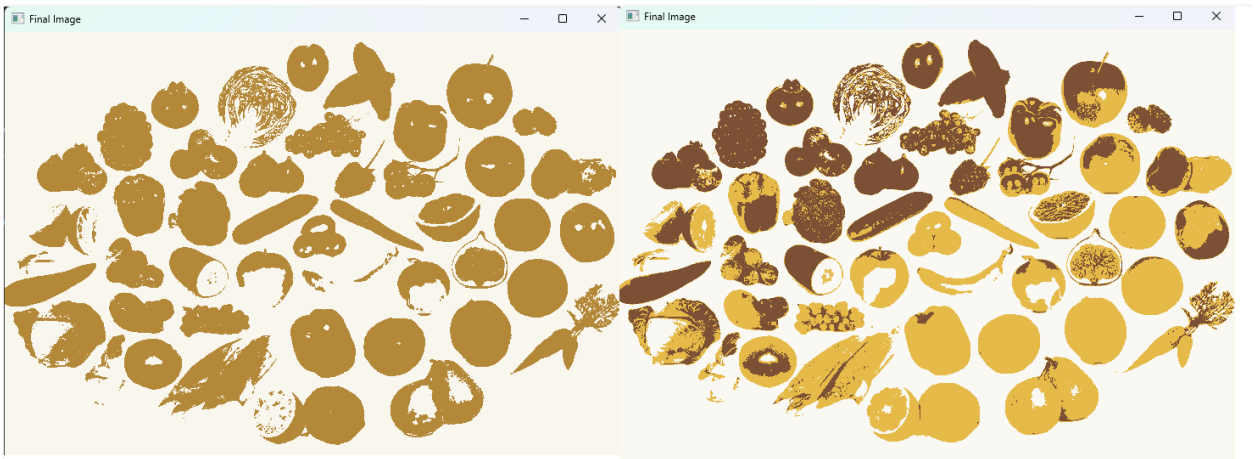
6. Финальное изображение сегментации создается на основе центров кластеров. С помощью массива labels, который содержит метки кластеров для каждого пикселя исходного изображения мы получаем, что каждый пиксель изображения получает соответствующий цвет из массива centers в зависимости от своей метки.

Затем нам необходимо изменить форму итогового изображения, чтобы оно совпадало с исходным изображением. Массиву final_image, где каждому пикселю соответствует его цвет, мы присваиваем его же значение с размерами, соответствующими исходному изображению.

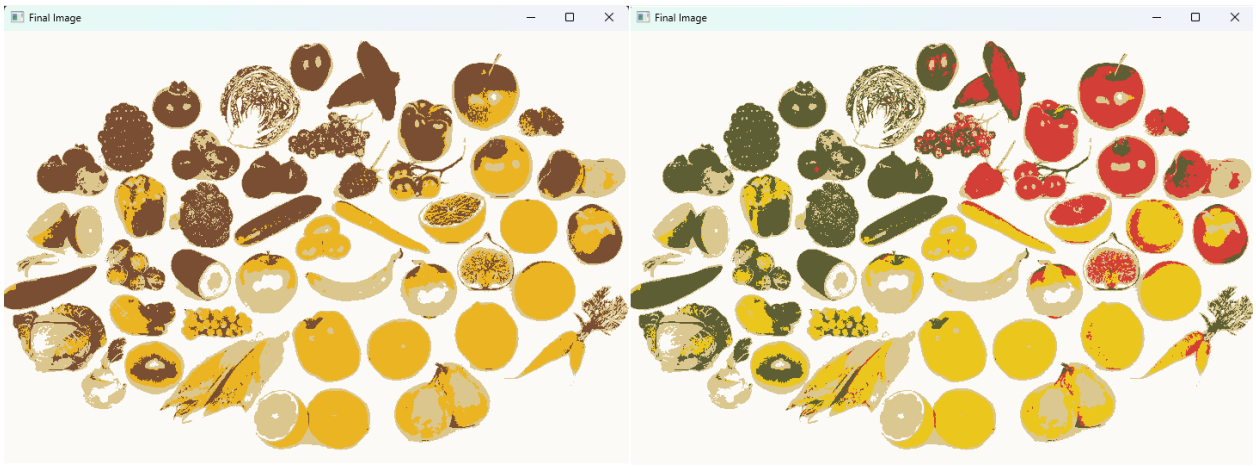
```
32    # создание итогового изображения, где каждый пиксель имеет цвет соответствующего кластера
33    final_image = centers[labels.flatten()]  # делаем из labels одномерный массив
34    final_image = final_image.reshape(image.shape)
```

Результаты отображения в зависимости от заданного количества кластеров:

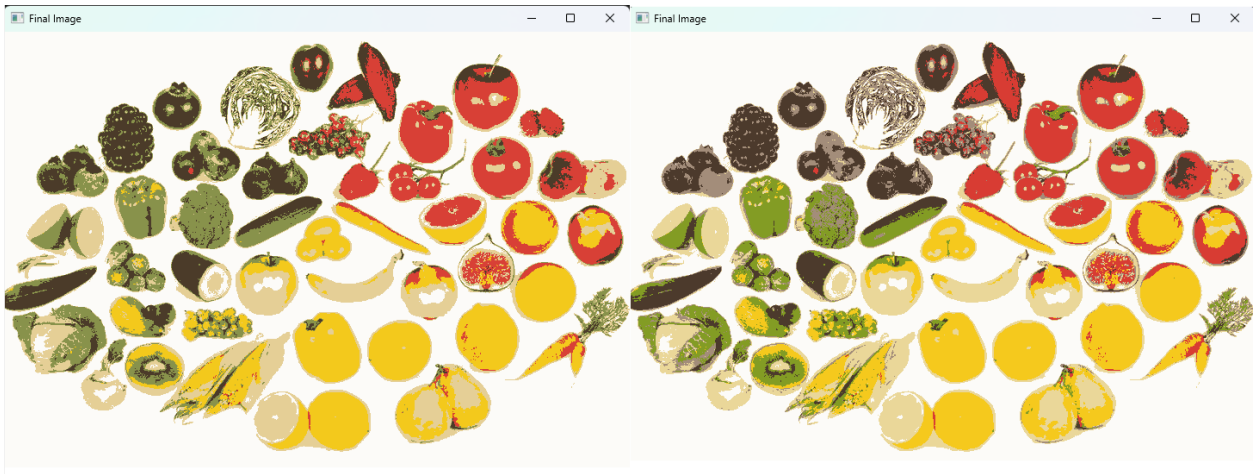
k = 2; k = 3;



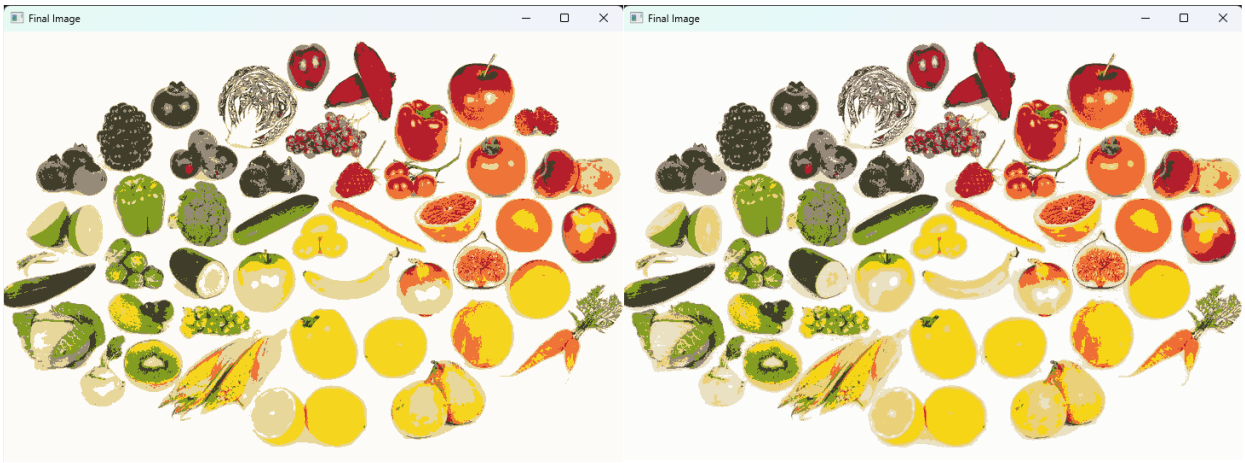
k = 4; k = 5;



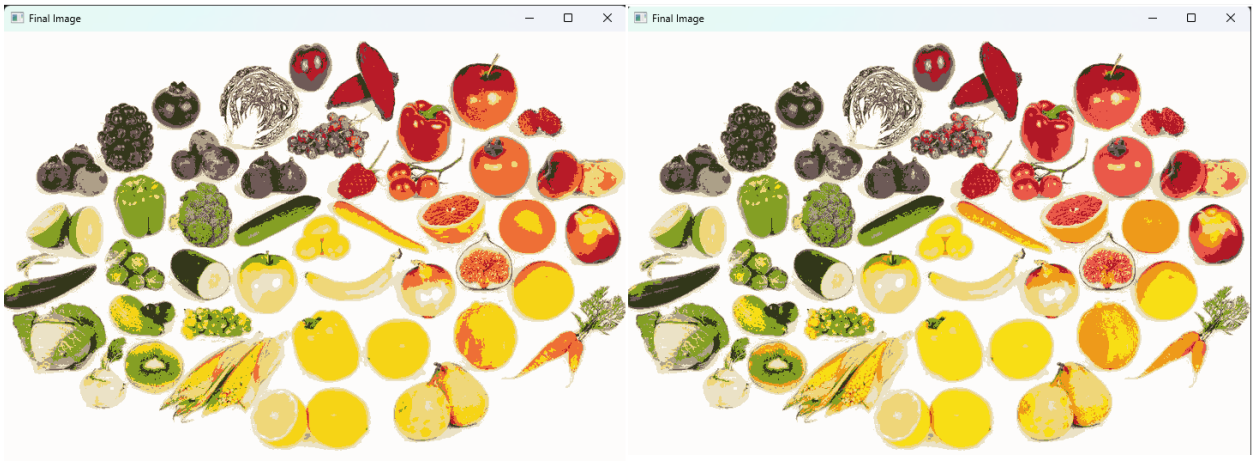
k = 6; k = 7;



k = 8; k = 9;



k = 10; k = 11.



Листинг кода на языке Python:

```python
import cv2
import numpy as np


def show_res(frame_name, res):
    cv2.imshow(frame_name, res)
    cv2.waitKey(0)
    cv2.destroyAllWindows()


# загрузка изображения
image = cv2.imread("D:/cv_projects/lab2/fruits_vegs.jpg")

show_res("Original Image", image)

# задаем количество кластеров
```

```
k = int(input("Введите количество кластеров (k): "))

# приведение формы изображения к двумерному массиву пикселей
pixel_values = image.reshape((-1, 3))
pixel_values = np.float32(pixel_values)  # преобразование во float

# определение критериев завершения и выполнение алгоритма k-средних
criteria_end = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)
_, labels, centers = cv2.kmeans(
    pixel_values, k, None, criteria_end, 10, cv2.KMEANS_RANDOM_CENTERS
)

# преобразование центров обратно в uint8
centers = np.uint8(centers)

# создание итогового изображения, где каждый пиксель имеет цвет соответствующего
кластера
final_image = centers[labels.flatten()]  # делаем из labels одномерный массив
final_image = final_image.reshape(image.shape)

# отображение итогового изображения
show_res("Final Image", final_image)
```