In [1]:
```python
import cv2
import numpy as np
import os
import struct
```

In [2]:
```python
class ImagePreparator:
    def read_image(filepath: str) -> np.ndarray:
        image = None
        if filepath.endswith('.bin'):
            image = ImagePreparator.read_bin(filepath)
        else:
            image = cv2.imread(filepath, cv2.IMREAD_GRAYSCALE)
        return image

    def read_bin(filepath: str) -> np.ndarray:
        with open(filepath, 'rb') as f:
            height = struct.unpack('I', f.read(4))[0]
            width = struct.unpack('I', f.read(4))[0]
            data = f.read()
            image = np.frombuffer(data, dtype=np.uint8).reshape(height, width)
        return image

    def save_bin(image: np.ndarray, filepath: str):
        with open(filepath, 'wb') as f:
            height, width = image.shape
            f.write(struct.pack('II', height, width))
            f.write(image.tobytes())

    def get_file_size(filepath: str) -> int:
        return os.path.getsize(filepath)
```

In [3]:
```python
def haar_1d_vectorized(data: np.ndarray) -> np.ndarray:
    data_f16 = data.astype(np.float16)
    n = len(data_f16)

    even = data_f16[::2] # каждый второй с 0
    odd = data_f16[1::2] # каждый второй с 1

    avg = (even + odd) * 0.5
    diff = (even - odd) * 0.5

    result = np.zeros(n, dtype=np.float16)
    result[:n // 2] = avg  # первая половина - среднее
    result[n // 2:] = diff # вторая половина - разности

    return result


def haar_2d(image: np.ndarray) -> tuple:
    image = ensure_even_size(image)
    h, w = image.shape

    temp = np.zeros_like(image, dtype=np.float16)
```

```
    result = np.zeros_like(image, dtype=np.float16)

    for i in range(h):
        temp[i, :] = haar_1d_vectorized(image[i, :])

    for j in range(w):
        result[:, j] = haar_1d_vectorized(temp[:, j])

    h2, w2 = h // 2, w // 2
    LL = result[:h2, :w2]  # Низкие частоты
    LH = result[:h2, w2:]  # Вертикальные детали
    HL = result[h2:, :w2]  # Горизонтальные детали
    HH = result[h2:, w2:]  # Диагональные детали

    return LL, LH, HL, HH

def ensure_even_size(image: np.ndarray) -> np.ndarray:
    h, w = image.shape
    new_h = h if h % 2 == 0 else h - 1
    new_w = w if w % 2 == 0 else w - 1
    return image[:new_h, :new_w]
```

In [4]:
```
def quantize(source_array: np.ndarray, levels: int = 4):
    step = (source_array.max() - source_array.min()) / levels

    boundaries = [source_array.min() + i * step for i in range(levels + 1)]
    averages = [boundaries[i] for i in range(levels)]

    quantized = np.zeros_like(source_array)
    for i in range(levels):
        if i == levels - 1:
            mask = (source_array >= boundaries[i]) & (source_array <= boundari
        else:
            mask = (source_array >= boundaries[i]) & (source_array < boundarie
        quantized[mask] = averages[i]

    return quantized
```

In [5]:
```
def run_length_encode(matrix):
    flattened = matrix.flatten()

    encoded = []
    current_value = flattened[0]
    count = 1

    for value in flattened[1:]:
        if abs(value - current_value) < 10e-6:
            count += 1
        else:
            encoded.append((current_value, count))
            current_value = value
            count = 1
```

```
    encoded.append((current_value, count))
    return encoded


def save_haar_compressed(LL, LH, HL, HH, filename):
    with open(filename, 'wb') as f:
        h, w = LL.shape
        f.write(struct.pack('fI', h, w))
        f.write(LL.astype(np.float16).tobytes()) # float16 для не потери вещес

        for component in [LH, HL, HH]:
            unique_values, counts = np.unique(component, return_counts=True)
            f.write(struct.pack('B', len(unique_values))) # информация, скольк
            for value, count in zip(unique_values, counts):
                f.write(struct.pack('fI', value, count)) # запись пар
```

In [6]:

```
'''
 Задача №1 : сохранение исходного изображения JPG -> BIN
'''
source_image = ImagePreparator.read_image("sar_1_gray.jpg")
#source_image = ImagePreparator.read_image("cells.jpg")
ImagePreparator.save_bin(source_image, "original_image.bin")

jpg_size = ImagePreparator.get_file_size("sar_1_gray.jpg")
#jpg_size = ImagePreparator.get_file_size("cells.jpg")
bin_size = ImagePreparator.get_file_size("original_image.bin")

print("\n" + "=" * 50)
print("СКАЧИВАНИЕ ИСХОДНОГО ИЗОБРАЖЕНИЯ")
print("=" * 50)
print(f"JPG (сжатый): {jpg_size} байт")
print(f"BIN (несжатый): {bin_size} байт")
```

```
==================================================
СКАЧИВАНИЕ ИСХОДНОГО ИЗОБРАЖЕНИЯ
==================================================
JPG (сжатый): 71132 байт
BIN (несжатый): 240008 байт
```

In [7]:

```
'''
 Задача №2 : вейвлет-преобразование Хаара
'''
LL, LH, HL, HH = haar_2d(source_image)

print("\n" + "=" * 50)
print("РЕЗУЛЬТАТ ПРЕОБРАЗОВАНИЯ ХААРА")
print("=" * 50)
print(f"LL (приближение): {LL.shape}")
print(f"LH (вертикальные детали): {LH.shape}")
print(f"HL (горизонтальные детали): {HL.shape}")
print(f"HH (диагональные детали): {HH.shape}")

print("\n" + "=" * 50)
print("ПРИМЕРЫ ДАННЫХ (первые 10x10 элементов):")
```

```
print("=" * 50)

print("LL (Низкие частоты - приближение):")
print(LL[:10, :10].round(2))

print("\nLH (Вертикальные детали):")
print(LH[:10, :10].round(2))

print("\nHL (Горизонтальные детали):")
print(HL[:10, :10].round(2))

print("\nHH (Диагональные детали):")
print(HH[:10, :10].round(2))

print("\n" + "=" * 50)
print("ДИАПАЗОНЫ ЗНАЧЕНИЙ:")
print("="*50)
print(f"LL: [{LL.min():.2f}, {LL.max():.2f}]")
print(f"LH: [{LH.min():.2f}, {LH.max():.2f}]")
print(f"HL: [{HL.min():.2f}, {HL.max():.2f}]")
print(f"HH: [{HH.min():.2f}, {HH.max():.2f}]")
```

```
==================================================
РЕЗУЛЬТАТ ПРЕОБРАЗОВАНИЯ ХААРА
==================================================
LL (приближение): (200, 300)
LH (вертикальные детали): (200, 300)
HL (горизонтальные детали): (200, 300)
HH (диагональные детали): (200, 300)

==================================================
ПРИМЕРЫ ДАННЫХ (первые 10x10 элементов):
==================================================
LL (Низкие частоты - приближение):
[[ 55.    59.53  52.25  55.    64.75  85.75  63.    73.25  78.25  89.25]
 [ 57.    34.    43.25  62.75  57.53  70.75  73.25  79.25  76.5   70.75]
 [ 51.25  52.25  65.25  64.75  52.25  62.25  56.25  87.5   79.   100.94]
 [ 71.    65.25  56.75  67.    76.5   74.    66.5   61.    83.06  69.75]
 [ 52.    48.25  60.25  69.25  65.25  79.5   78.    76.25  60.47  62.  ]
 [ 58.75  47.53  45.53  42.25  54.    56.25  74.   133.8   90.25 118.75]
 [ 52.75  48.25  51.53  48.47  52.    78.75  89.75 100.5   94.5   86.25]
 [ 41.    43.53  53.75  52.75  67.75  70.    73.5   50.75  72.75  74.  ]
 [ 52.47  59.25  61.    66.25  57.    76.5   78.5   73.    70.   105.75]
 [ 71.25  48.75  79.25  85.75  77.25  96.5  116.5   93.25 101.25 116.5 ]]

LH (Вертикальные детали):
[[ -6.     5.    -2.75   2.5   11.25 -12.75  13.   -10.25   8.75  -6.25]
 [  4.5    4.5   -7.75  -1.25   4.5   -5.25  -3.75   5.75   0.5   -8.25]
 [  2.75  -4.25  -0.75  -0.25  -0.25   1.75  -9.25   2.    -8.   -10.  ]
 [  4.    -0.75   4.25  -7.5   11.5   -5.     2.5    5.    16.5   -4.75]
 [ -1.     1.75  -6.75   2.25  -8.25   2.5   -3.     4.25  20.5    3.  ]
 [ -0.25   4.5   -3.     4.25   3.    -4.25  -7.   -18.25  10.25  -1.75]
 [ -1.25   2.75  -3.5    4.5   -1.5   -9.75   2.75  -7.5   -9.    21.23]
 [ -2.5    0.5   -4.75   4.25  -3.75   2.    -2.    10.25 -14.75  19.  ]
 [ -3.5    0.75  -2.     0.25   7.5  -10.     5.    -4.     8.     1.25]
 [  6.75   0.25 -10.75   5.25  13.75  -2.5    2.5  -25.77   6.75  -1.  ]]

HL (Горизонтальные детали):
[[-20.5    4.     8.75  -2.   -10.75   6.75   8.5   11.75  26.23   1.25]
 [ 12.5    0.    -7.25   1.25  -7.5   -9.25  -1.75   6.75 -14.5   -0.25]
 [ -0.25   0.75   1.25  -6.25   8.25  12.25   6.75  -8.5   -2.5   -2.  ]
 [-15.   -12.75  -0.25   8.     4.5   -7.5  -18.5   -4.    12.     8.25]
 [ -8.    -7.75  -6.25  -5.25   2.75  -5.    -2.5    3.25   0.     4.5 ]
 [ -0.25   2.     6.5   10.25   2.5    4.25  -1.   -12.75  -2.25  -2.75]
 [ 10.75   5.75   0.    -3.5   -3.     1.25   1.75   9.     4.5   -0.25]
 [-11.    -9.5   -7.25  -4.75  -2.75  -8.     0.    25.23 -12.25   4.  ]
 [  1.5    2.25 -16.5  -17.75  -7.5   -6.5  -16.   -18.5  -13.   -28.77]
 [-19.75 -11.25  -5.75 -10.25  -7.75  -4.     2.     9.75 -13.25  18.  ]]

HH (Диагональные детали):
[[-5.5  -4.5   1.75  2.5  -2.25 -4.75  2.5  -3.75  6.75  7.75]
 [ 1.    4.5  -1.25 -1.75  3.5  -3.25  0.25 -3.75  0.5  -3.25]
 [-0.75  0.25  0.25  2.75 -2.25  0.75  2.25  4.   -2.5  -1.  ]
 [ 0.   -1.75 -3.75 -0.5  -0.5   6.5  -2.5  -3.   -3.5   2.75]
 [ 0.   -0.25 -0.25 -0.25  0.25  3.   -3.5   1.25 -3.    1.5 ]
 [-0.25 -1.   -1.   -0.75 -2.5   1.75  1.    4.25  5.75 -3.25]
```

```
[ 0.75  1.25  1.    0.5   0.5  -2.25  0.75 -4.    6.   -1.25]
[-0.5  -0.5  -0.75 -0.25  2.75 -1.   -3.5  -7.25 -4.75 -2.  ]
[ 0.5   0.75  6.5  -4.75 -2.   -1.    2.5   6.5   8.   -2.25]
[-5.25  1.25 -3.75  5.25 -1.25 -3.   -1.    1.75 -8.75  2.5 ]]

==================================================
ДИАПАЗОНЫ ЗНАЧЕНИЙ:
==================================================
LL: [0.00, 254.50]
LH: [-86.00, 97.50]
HL: [-90.50, 89.50]
HH: [-67.00, 53.25]
```

In [8]:
```
'''
 Задача №3 : квантование с количеством квантов = 4
'''
LH_quantized = quantize(LH)
HL_quantized = quantize(HL)
HH_quantized = quantize(HH)
```

In [9]:
```
'''
 Задача №4 : сохранение результирующего изображения
'''
save_haar_compressed(LL, LH_quantized, HL_quantized, HH_quantized, "compressed
```

In [10]:
```
'''
 Итог : сравнение исходного и сжатого файлов
'''
print("\n" + "="*50)
print("СРАВНЕНИЕ ФАЙЛОВ:")
print("="*50)

compressed_size = ImagePreparator.get_file_size("compressed_haar.bin")
size_difference = bin_size - compressed_size

print(f"Исходный BIN:  {bin_size} байт")
print(f"Сжатый Haar:   {compressed_size} байт")
print(f"Разница:       {size_difference} байт")

if size_difference > 0:
    print(f"Экономия:      {size_difference} байт")
    print(f"Сжатие:      {(size_difference/bin_size)*100:>11.1f}%")

print(f"\nРазмер compressed_haar.bin составляет {compressed_size/bin_size*100:
```

```
==================================================
СРАВНЕНИЕ ФАЙЛОВ:
==================================================
Исходный BIN:  240008 байт
Сжатый Haar:   120107 байт
Разница:       119901 байт
Экономия:      119901 байт
Сжатие:             50.0%

Размер compressed_haar.bin составляет 50.0% от original_image.bin
```

In [ ]: