Each screen has pixels, which contain transistors with three color lights (red, green, blue), allowing us to see colors. But how do we tell each pixel which color to display? We achieve this using bits. As we have learned, bits can represent 2^N states, where N is the number of binary digits. Therefore, the solution is to encode each color with their help. For example, we can say black=0 and white=1. Thus, the computer reads the binary values, understands the color thanks to the coding, sends it to the pixels, and ultimately, I see the image.

So, to find the values that the pixels take, I will examine all of them and store all the different values once each in a list.

Discrete source symbols:

In the given image, I execute the command:

```
I = imread('parrot.png');
```

And I observe that a variable appears in the workspace.

The uint8 means unsigned 8-bit integer, so each pixel consists of 8 bits. Therefore, each pixel can take 256 different values. Normally, each RGB color is described by 8 bits, resulting in a total of 24 bits per pixel. However, since the image is black and white, there is a convention that all three colors take the same values. Thus, 8 bits are enough for grayscale images.

Therefore, theoretically, I will have 256 discrete values. This does not mean that all available values will be used; some of them are enough for an image.

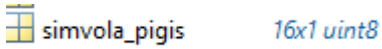
So, let's find out which ones are used for the formation of the image.

We use the function:

```
simvola_pigis = unique(I(:));
```

to see which values the pixels have actually taken.

This is generated



and we know that there are finally 16 discrete values that the pixels take.

To display the exact values, I use the command:

```
disp(simvola_pigis);
```

and I get

```
  0
 17
 34
 51
 68
 85
102
119
136
153
170
187
204
221
238
255
```

Probability of occurrence :

After identifying the source symbols, now we need to find their probabilities of occurrence, meaning which values the pixels take more often and which less. To calculate them, we simply say: number of pixels with a specific value / total number of pixels in the image. Thus, we use the command:

```
pithanotites = histcounts(I, numel(simvola_pigis)) / numel(I);
```

The histcounts records how many pixels have each discrete value, and numel(I) counts how many pixels the entire image has. Then, by dividing, we find the probability of occurrence of each symbol.

Results of executing the command :

```
pithanotites = histcounts(I, numel(simvola_pigis)) / numel(I);
disp(pithanotites);
```

```
0.0962    0.0814    0.0683    0.0625    0.0768    0.0905    0.1132    0.0906    0.0965    0.0671    0.0389    0.0329    0.0337    0.0259    0.0224    0.0031
```

Finally, we observe that if we add all the probabilities together, they sum to 1, meaning that the source indeed takes 16 discrete values.