

# FUNDAMENTAÇÃO TEÓRICA PARA O T1 – MANIPULAÇÃO DE IMAGENS

## O que é um arquivo PPM?

PPM é a abreviação de Portable Pixmap Format. Embora esses arquivos sejam raros hoje em dia, você pode identificá-los pela extensão .PPM.

O formato PPM surgiu no final dos anos 80 para facilitar o compartilhamento de imagens entre diferentes plataformas. Cada arquivo PPM usa um formato de texto para armazenar informações sobre uma determinada imagem. Em cada arquivo, cada pixel tem um número específico (de 0 a 65536) e informações sobre a altura e largura de uma imagem, além de dados do espaço em branco.

**P3**

# P3 means colors are in ASCII

# 3 columns

# 2 rows

# 255 for max color

# The file ends with 6 RGB triplets (18 integer values in ASCII)

3 2

255

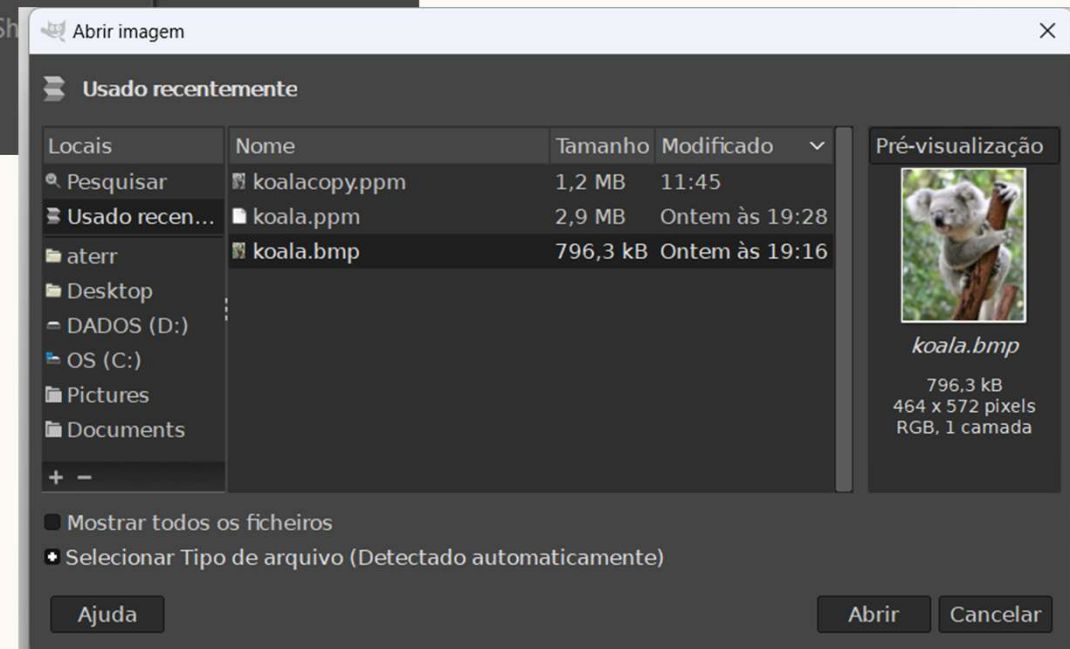
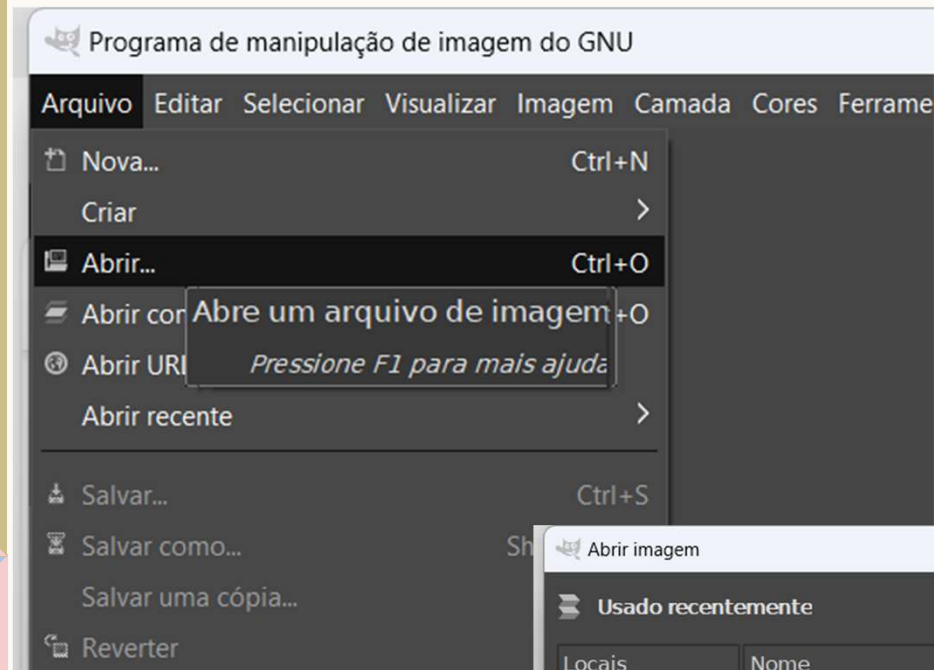
255 0 128 32 255 0 64 32 255 25 55 10 255 11 5 0 0 0

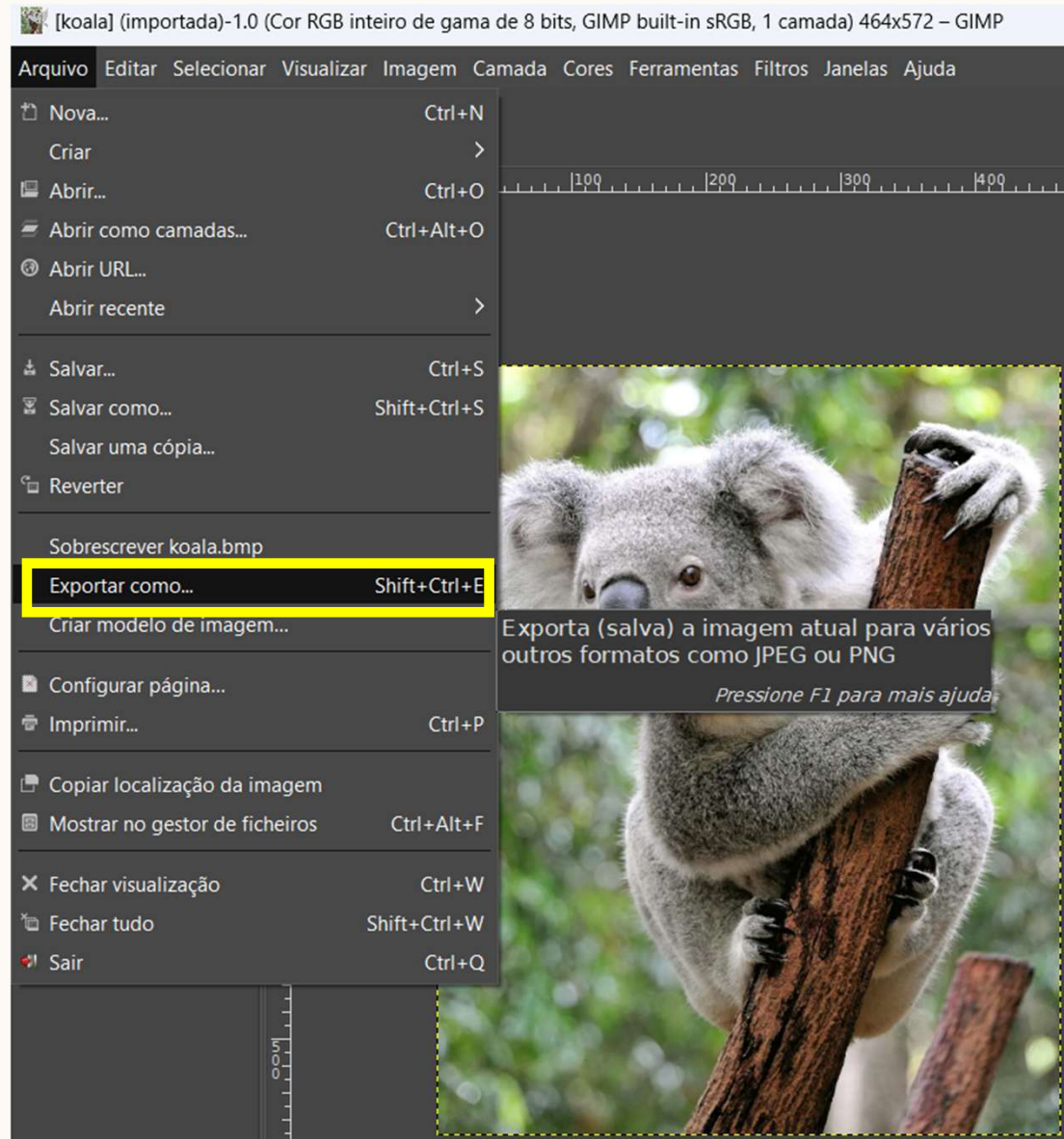
# GERANDO IMAGENS COM O GIMP<sup>10</sup>

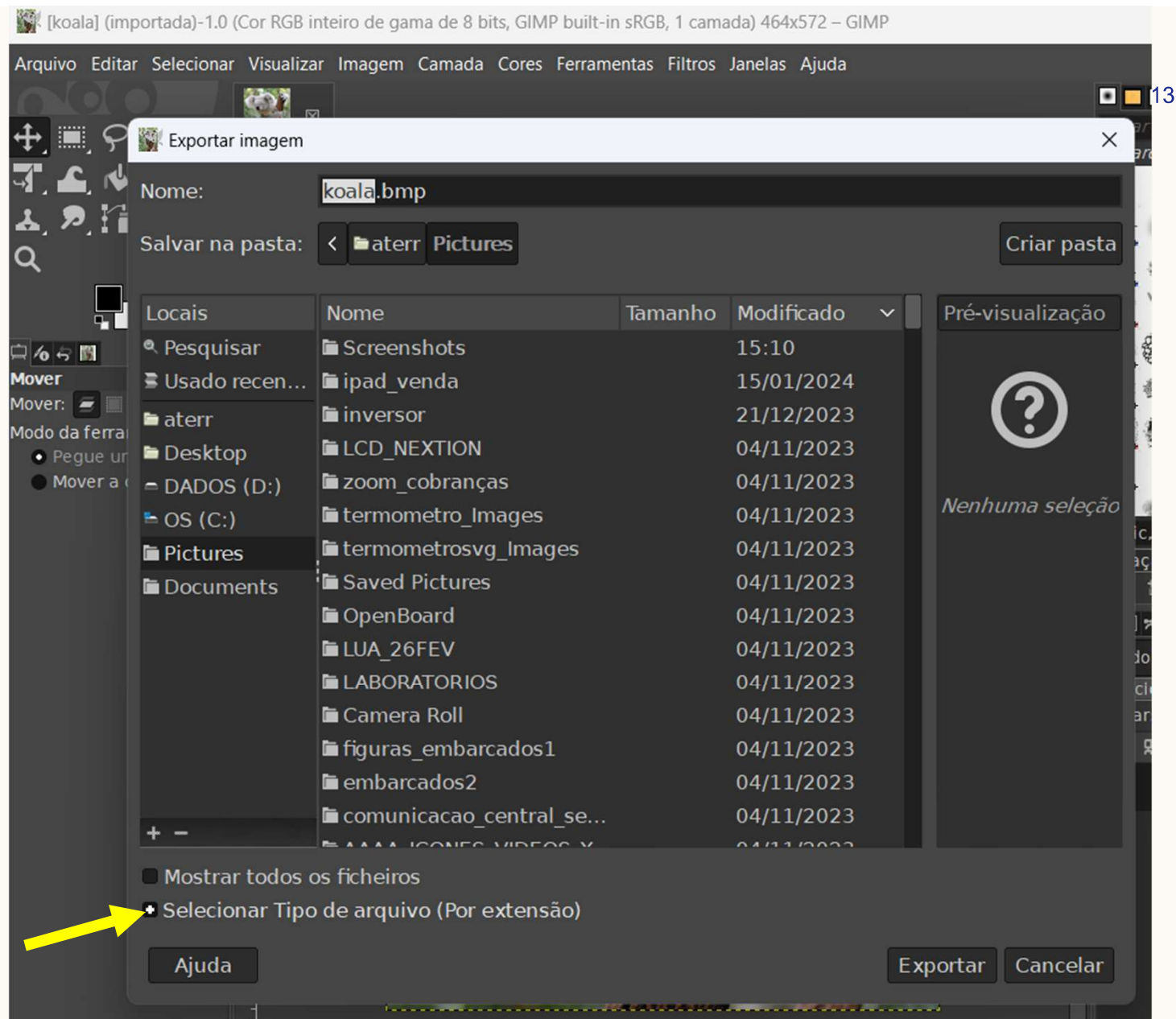
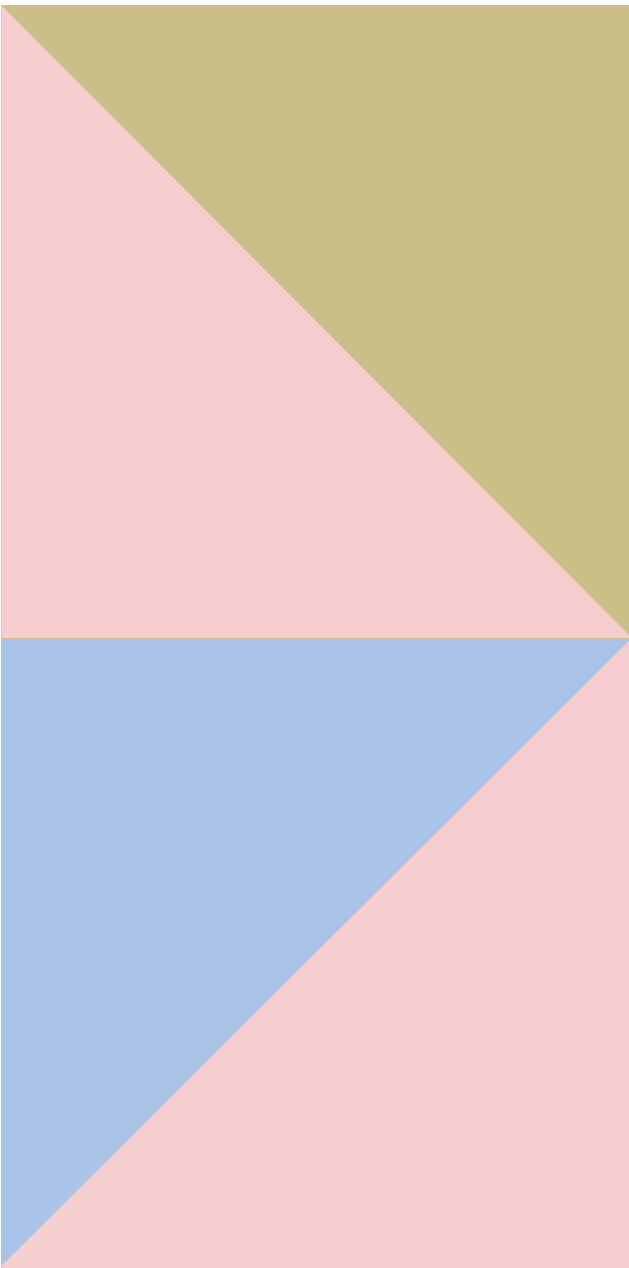
<https://www.gimp.org/downloads/>

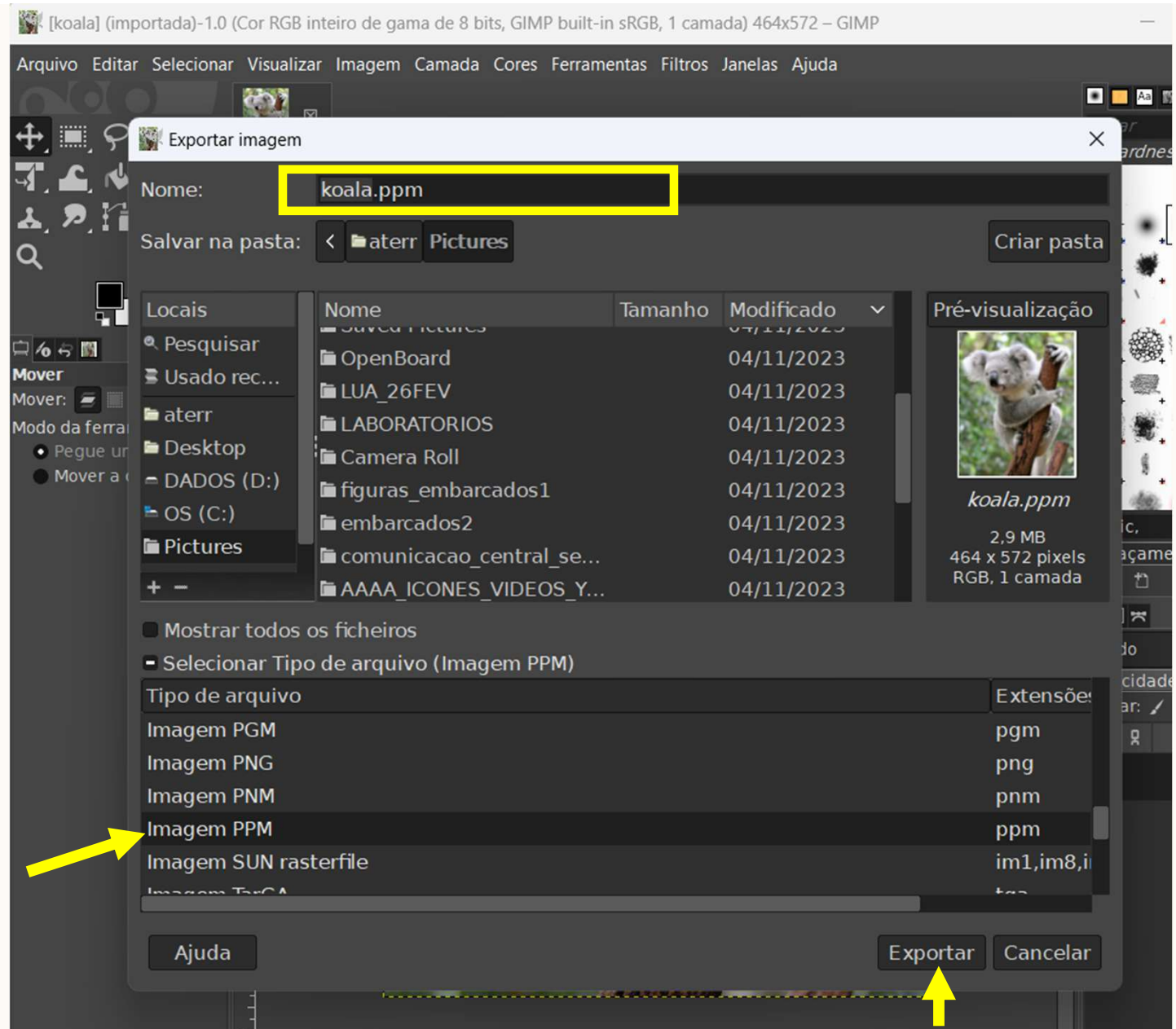


O GNU Image Manipulation Program permite salvar imagens no formato ppm (

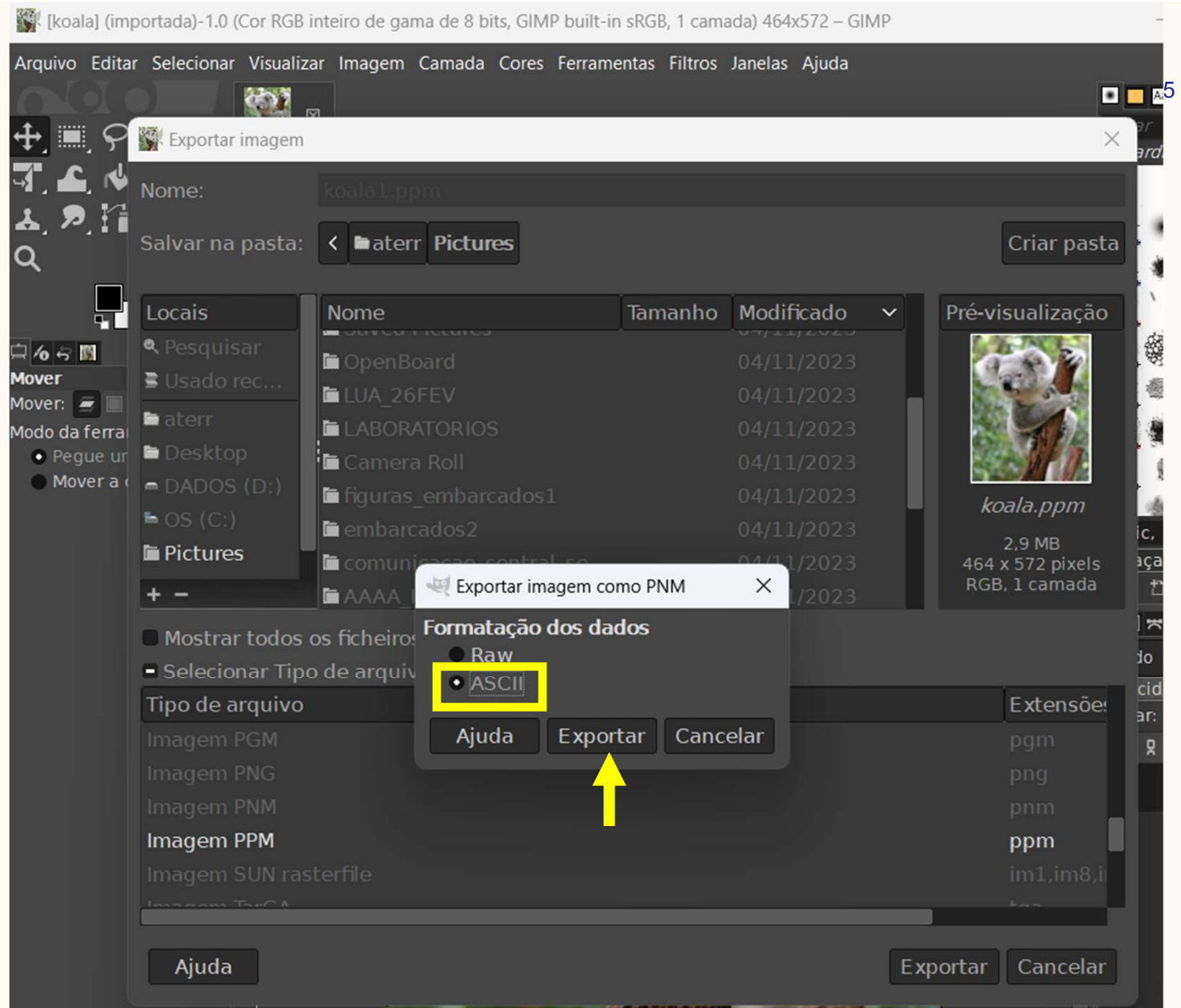












# ESTE ARQUIVO PODE SER ABERTO NO NOTEPAD, POR EXEMPLO.

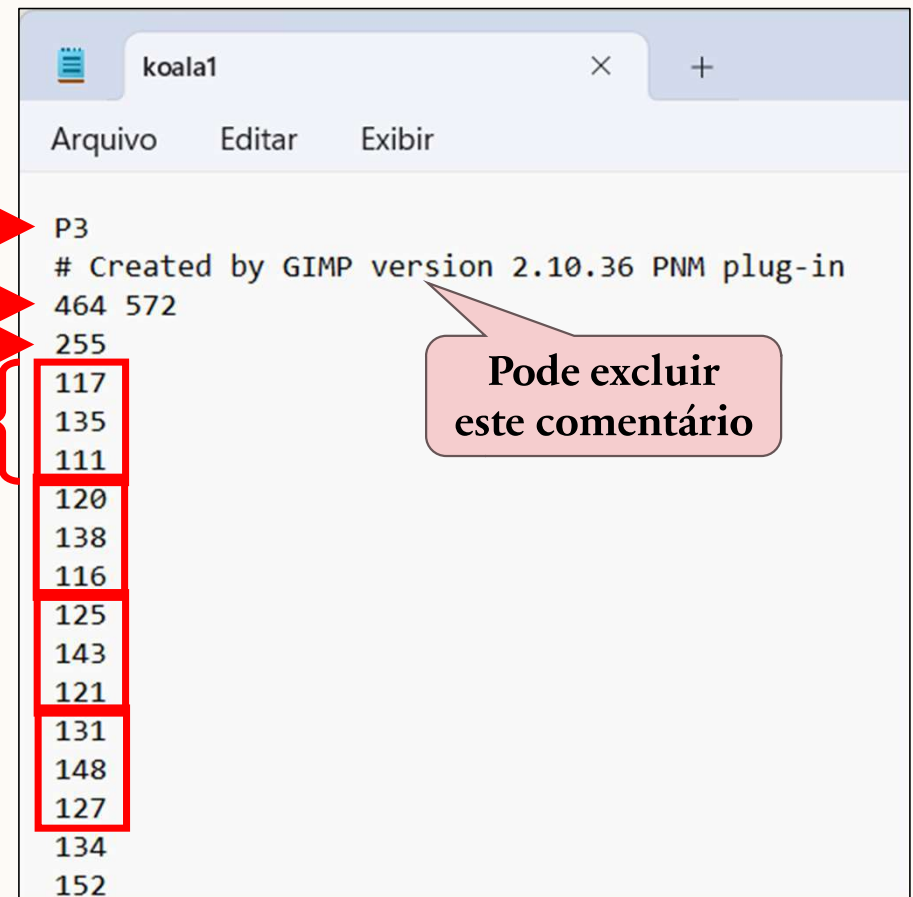
16

Formato ASCII e colorido

Tamanho da imagem

Valor máximo do pixel

Valores de R, G e B  
de cada pixel da imagem



```
P3
# Created by GIMP version 2.10.36 PNM plug-in
464 572
255
117
135
111
120
138
116
125
143
121
131
148
127
134
152
```

Pode excluir este comentário



## Manipulação de arquivos:

1) Abertura de um arquivo – como exemplo usaremos a imagem de um koala.

17

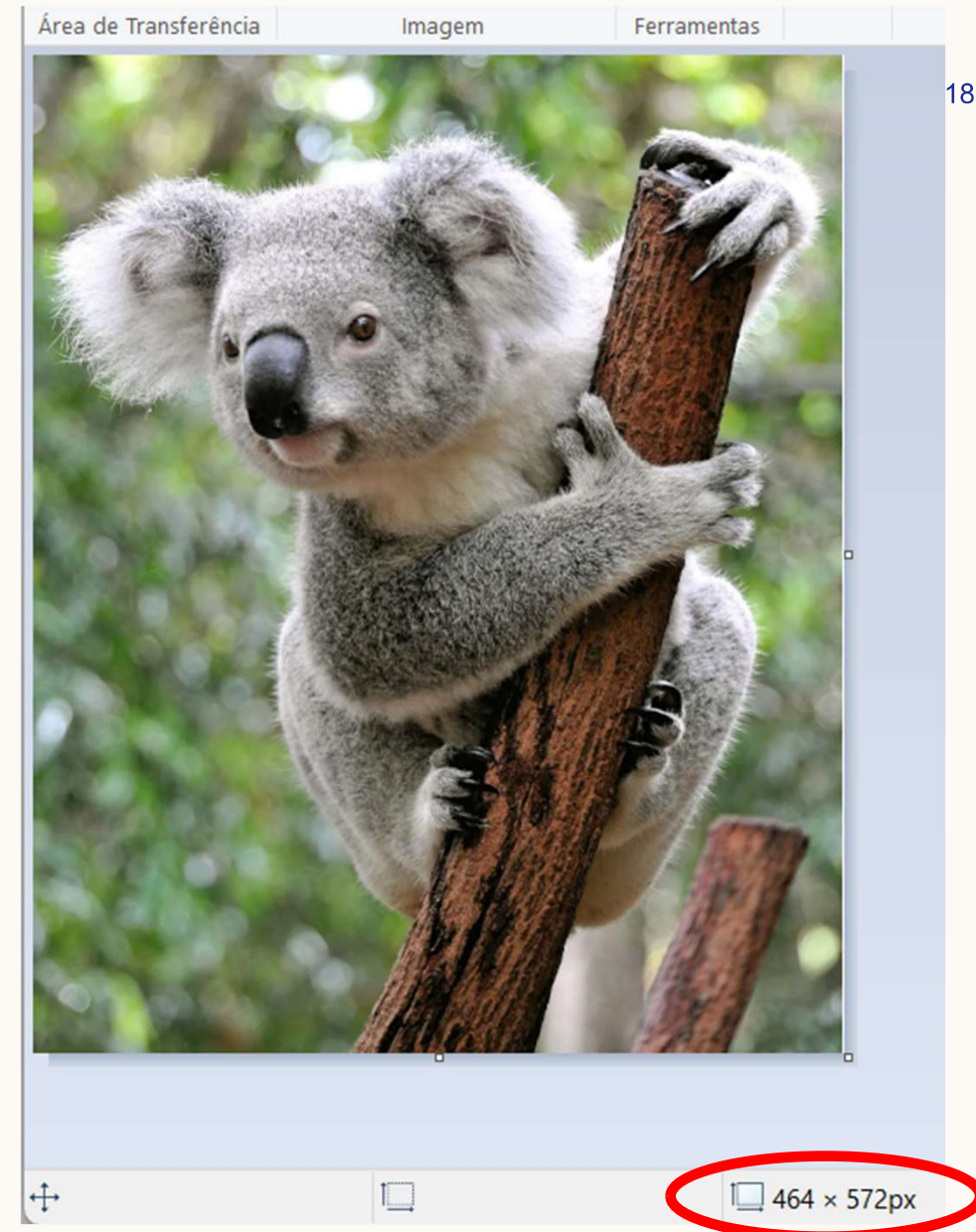


Esta imagem tem uma resolução de 464 x 572 pixels.

São 572 linhas x 464 colunas.

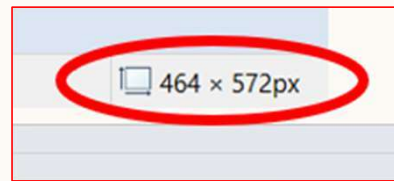
Cada pixel é representado por 3 bytes – Red, Green, Blue.

Cada byte varia de 0 a 255.



# VERIFICAÇÃO DOS PIXELS DA IMAGEM

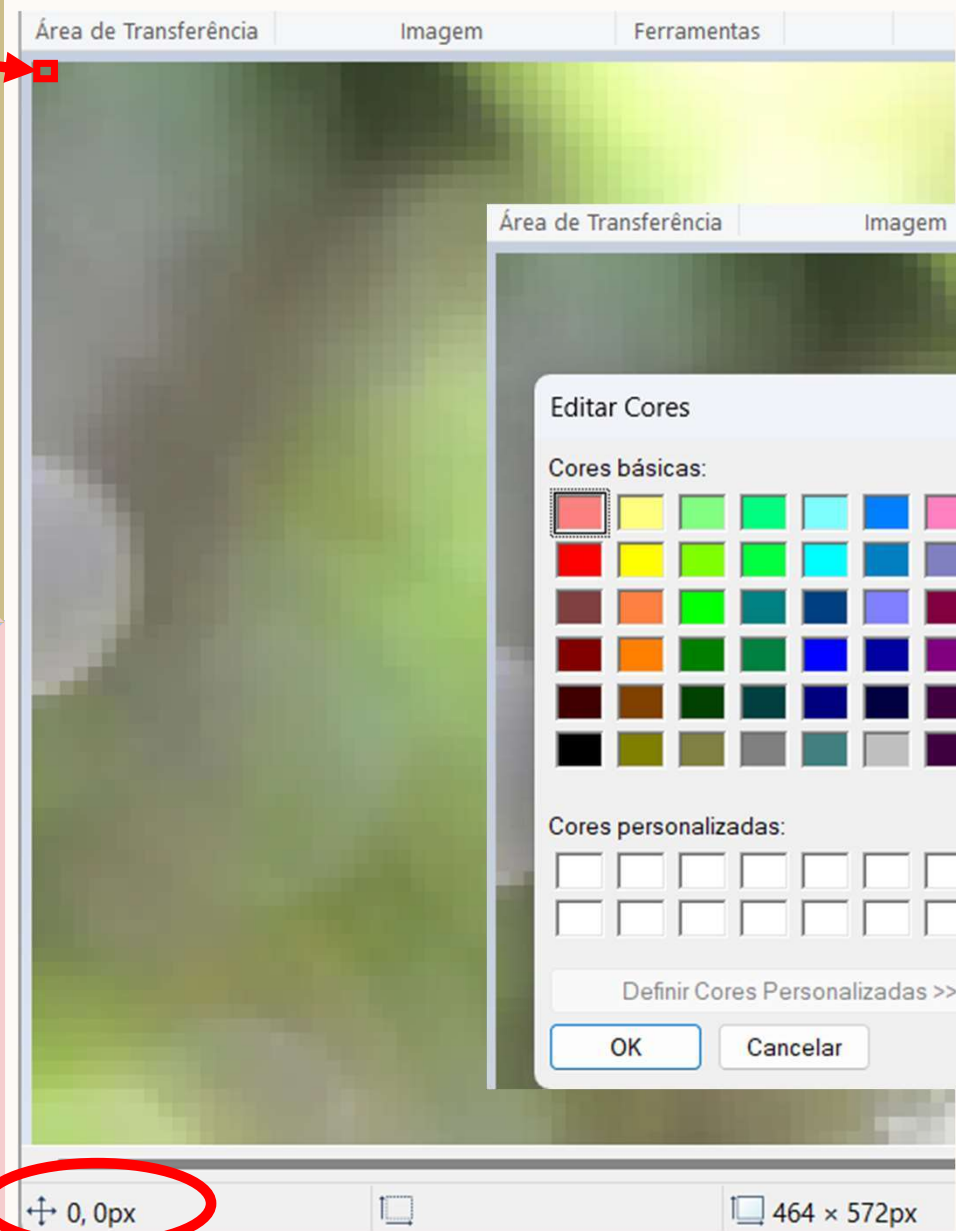
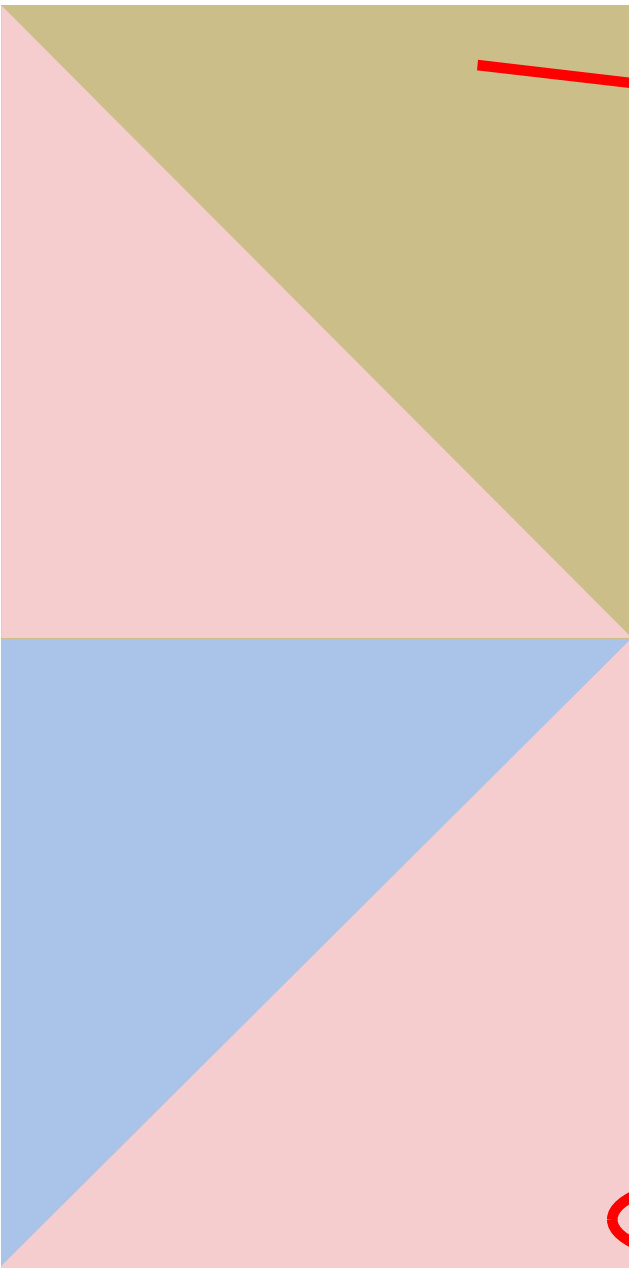
19



```
Arquivo  Editar  Exibir

P3
# Created by GIMP version 2.10.36 PNM plug-in
464 572
255
117
135
111
120
138
116
```

Nos próximos dois *slides* mostra o valor do pixel (0,0) e (1,0) da imagem num editor e compara-se o valor do pixel no arquivo ppm.



**PIXEL (0,0)** 20

Arquivo Editar

P3  
# Created by GIMP

|     |     |
|-----|-----|
| 464 | 572 |
| 255 |     |
| 117 |     |
| 135 |     |
| 111 |     |
| 120 |     |
| 138 |     |
| 116 |     |

Editar Cores

Cores básicas:

|  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Cores personalizadas:

|  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Definir Cores Personalizadas >>

OK Cancelar

Cor | Sólida

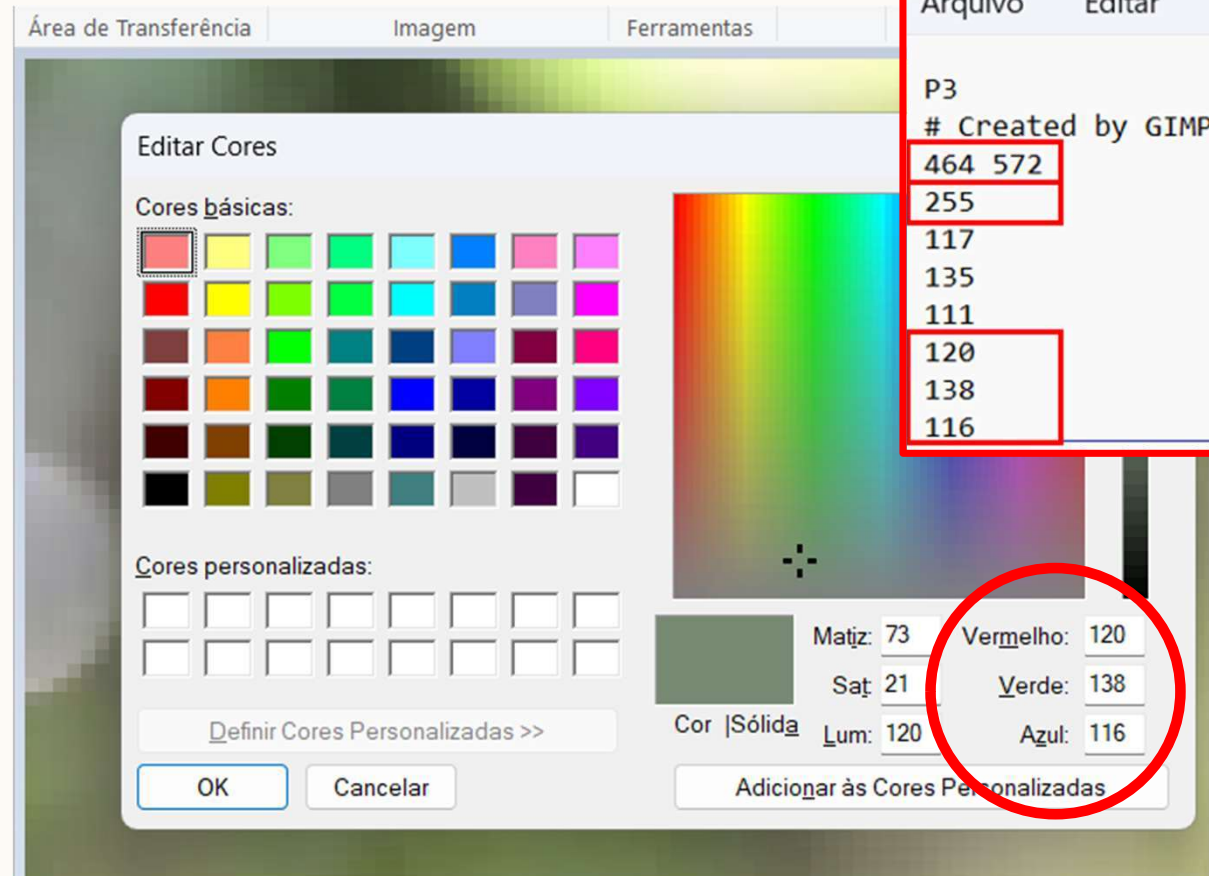
Matiz: 70  
Sat: 23  
Lum: 116

Vermelho: 117  
Verde: 135  
Azul: 111

Adicionar às Cores Personalizadas



**PIXEL (1,0)**





## ABERTURA DE ARQUIVO

```
FILE *fp;  
fp = fopen("koala.ppm", "r");
```

## LEITURA DO FORMATO

```
char formato[3];  
fscanf(fp, "%s", formato);    // lê o tipo de imagem P3 (color), P2 (P&B)  
printf("%s\n", formato);
```

## LEITURA DO TAMANHO DA IMAGEM

```
int coluna, linha;  
fscanf(fp, "%d %d", &coluna, &linha);    // lê o tamanho da matriz  
printf("%d %d\n", coluna, linha);
```

## LEITURA DO VALOR MÁXIMO POR PIXEL

```
int valor;  
fscanf(fp, "%d", &valor);    // lê o valor máximo.  
printf("%d\n", valor);
```



# **CRIANDO UM ARQUIVO PPM**

## **ABERTURA DE ARQUIVO**

```
FILE* fp_novo = fopen ("koalacopy.ppm", "w");
```

## **ESCREVENDO O CABEÇALHO NO ARQUIVO**

```
fprintf (fp_novo, "P2\n");  
fprintf (fp_novo, "%d %d\n", coluna, linha);  
fprintf (fp_novo, "%d\n", valor);
```

## **PRÓXIMO PASSO ESCREVER A MATRIZ DE PIXEL'S – PODE-SE USAR STRUCT.**

## LEITURA DE TODOS OS PIXELS DA IMAGEM

```
int i, j;

for(j=0; j<linha; j++)
{
    for(i=0; i<coluna; i++)
    {
        fscanf(fp, "%d %d %d", &r, &g, &b);
        printf("%d %d %d\n", r, g, b);
    }
}
```

## FECHAMENTO DO ARQUIVO

```
fclose(fp);
```

# COMO CRIAR UMA STRUCT DOS PIXELS 25

```
#include <stdio.h>
```

```
typedef struct  
{  
    int R;  
    int G;  
    int B;  
} RGB;
```

Esta struct é uma estrutura composta por 3 números inteiros – neste caso R, G e B.

```
int main()  
{
```

```
    RGB vetor[3][3];  
    vetor[0][0].R = 130;  
    vetor[0][0].G = 150;  
    vetor[0][0].B = 120;  
    printf("%d - %d - %d", vetor[0][0].R, vetor[0][0].G, vetor[0][0].B);
```

```
    return 0;
```

```
}
```

Aqui é feita a instanciação da struct RGB do vetor bidimensional

A atribuição de um valor a matriz bidimensional, utiliza-se o ponto (.) com a instância a cada elemento da struct.