



Universidade Presbiteriana Mackenzie

ALGORITMOS E PROGRAMAÇÃO II

Introdução à compactação de imagens

Projeto 2 (Proj 2)

(Atividade em grupo: 3 alunos por grupo)

Jefferson Zanutto

Leonardo Takuno

Contextualização

Trocas informações é uma tarefa essencial. No mundo digital isto ocorre de diversas formas, tipos de arquivos, textos, sons e imagens. Diversos são os métodos e algoritmos utilizados para compactação de informações. Na compactação de texto podemos citar o método de compressão mais usado utiliza Códigos de Huffman, aplicado na compressão ZIP e GZIP.

Há inúmeras aplicações para a compactação de imagens: redução do tempo de carregamento de páginas web; economizar espaço para armazenamento nos diversos meios disponíveis atualmente; etc.

Para a compressão de imagens o mais conhecido e utilizado é o JPEG que consegue compactar as imagens com perda de dados. Os arquivos de imagem PNG são, por sua vez, imagens compactadas, sem perda de dados.

Os arquivos BMP também são utilizados para representar imagens porém não são compactados e podem representar imagens com grande nitidez e precisão.

Objetivos do Projeto

O objetivo deste projeto é, em primeiro plano, a prática da atividade de programação através da solução do problema proposto. Aqui você poderá praticar os conceitos estudados na teoria da disciplina de Algoritmos e Programação II, através do desenvolvimento de uma aplicação capaz de compactar imagens no formato BMP.

Arquivos BMP


Apresentamos a seguir uma breve introdução aos arquivos Bitmap, direcionada a realização desta atividade.
















Uma imagem bitmap é, essencialmente, uma matriz de pixels de uma imagem. Há diversos padrões para representar bitmaps.

Para imagens em níveis de cinza, cada pixel da imagem pode ser representado utilizando, por exemplo, 4 bits para 16 níveis de cinza, 16 bits para 65536 níveis de cinza. Mais quantidade de bits, maior qualidade.

Para imagens coloridas, há diversas formas de representação. Uma imagem que utiliza 4 bits para armazenar cada pixel é capaz de representar $2^4 = 16$ cores diferentes que são indicadas em uma tabela de cores (contendo quais são as 16 cores). Nesta tabela, cada cor é representada utilizando 24 bits para compor o padrão RGB (8 bits para vermelho, 8 bits para verde, 8 bits para azul), conforme ilustrado abaixo.

3	3	3	3	3	3	3	3
0	1	4	1	4	1	4	0
0	4	1	4	1	4	1	0
0	5	5	5	5	5	5	0
0	5	5	5	5	5	5	0
0	1	4	1	4	1	4	0
0	4	1	4	1	4	1	0
2	2	2	2	2	2	2	2



0	000000	
1	FF0000	
2	00FF00	
3	0000FF	
4	FFFFFF	
5	FFFF00	
6	FF00FF	
7	00FFFF	
8	FF0080	
9	FF8040	
A	804000	
B	008080	
C	800000	
D	800080	
E	8080FF	

Entretanto, a representação interna de imagens bitmap mais comum utiliza 24 bits para representar cada pixel, sendo, portanto, capaz de representar uma gama de 2^{24} cores, ou seja, 16.777.216 de cores, também chamado de representação *true color*. Neste caso, não há a necessidade de utilização de uma tabela cores, pois cada pixel já está representado, na própria imagem, por um conjunto de 3 bytes (24 bits), 1 byte para R, 1 byte para G, 1 byte para B (RGB).

Neste trabalho você irá manipular apenas BMP *true color*, que utilizam 24 bits por pixel.

Os arquivos BMP puro é formado por um cabeçalho de 54 bytes seguido pela sequência de bytes que representam a imagem propriamente dita.

Uma imagem BMP é, como já mencionado, uma matriz de pixels, sendo que cada linha da imagem deve possuir uma quantidade de bytes múltipla de 4. Caso isto não ocorra, há um acréscimo ao final de cada linha a quantidade necessária de bytes de valor zero (00H) para que esta regra seja respeitada.

* 3 bytes = 183 bytes por linha, sendo necessário acrescentar em cada linha da imagem 1 byte zero, totalizando assim 184 bytes por linha de imagem (184 é divisível por 4).

Imagens BMP podem ainda ter um conjunto de bytes entre o cabeçalho da imagem e a imagem propriamente dita, dependendo de como a imagem foi gerada. Por exemplo, imagens geradas por aplicativos Windows.

A seguir há um exemplo que mostra os primeiros bytes de um arquivo BMP.

42	4D	CA	28	00	00	00	00	00	00	8A	00	00	00	7C	00
00	00	3D	00	00	00	38	00	00	00	01	00	18	00	00	00
00	00	40	28	00	00	C3	0E	00	00	C3	0E	00	00	00	00
00	00	00	00	00	00	00	00	FF	00	00	FF	00	00	FF	00
00	00	00	00	00	FF	42	47	52	73	8F	C2	F5	28	51	B8
1E	15	1E	85	EB	01	33	33	33	13	66	66	66	26	66	66
66	06	99	99	99	09	3D	0A	D7	03	28	5C	8F	32	00	00
00	00	00	00	00	00	00	00	00	00	04	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	E7	E7	E7	EF	EF	EF
F8	F8	F8	FE	FE	FE	FE	FE	FE	FD	FD	FD	FE	FE	FE	FF
FF	FF	FF	FF	FF	FF	FF	FF	FD	FD	FD	F8	F8	F8	FC	FC
FC	FF	FF	FF	FF	FF	FF	F9	F9	F9	FF	FE	FF	FF	FD	FF

CABEÇALHO WINDOWS

84 BYTES

CABEÇALHO
BMP DE 54
BYTES

IMAGEM

(aqui representada apenas parcialmente

Alguns destaques sobre o cabeçalho do arquivo BMP

Tamanho total do
arquivo .BMP em
bytes.

Quantidade de
colunas de pixels
em cada linha da
imagem

Quantidade de
linhas de pixels
da imagem

Imagem com 24 bits por pixel

Sempre. “BM”.
Iniciais de
BitMap.

Offset para o
início da
imagem

Neste exemplo, o arquivo BMP possui:

Tamanho total de 00 00 28 CA bytes = 10.442 bytes.

Offset para início da imagem: 00 8A bytes = 54 bytes

Quantidade de linhas de pixels: 56

Quantidade de pixels por linha: 61

Quantidade de bits por pixel: 24 (3 bytes).

Neste exemplo, cada linha da imagem é formada por $61(\text{pixels}) * 3(\text{bytes}) = 183 \text{ bytes}$.

Acrescido 1 byte 00H para completar 184 bytes por linha (184 é divisível por 4).

PROCEDIMENTO PARA A RELIZAÇÃO DA COMPACTAÇÃO

LEITURA DOS DADOS DO ARQUIVO

1º) Abrir um arquivo BMP 24 bits de nome **imagemOriginal.bmp**. Como não será realizado alocação dinâmica de memória neste projeto, então recomenda-se escolher arquivos com no máximo 11Kbytes.

2º) Ler o tamanho do arquivo em bytes.

2º) Ler a quantidade de linhas e colunas da imagem.

3º) Ler o offset indicado no cabeçalho

3º) Ler o cabeçalho do BMP e o “cabeçalho” extra eventualmente existente e que precede a representação da imagem e armazenar em um vetor de **unsigned char**. Este vetor será gravado junto com a matriz compactada posteriormente.

4º) Ler o arquivo BMP do início dos dados da imagem até o final e armazenar em uma matriz de **unsigned char**. Cada linha da imagem armazenada no arquivo corresponde a cada linha da matriz criada em memória.

No exemplo considerado anteriormente, o arquivo possui 61 pixels por linha. Logo cada linha da imagem é representada por uma sequência de $61 * 3 \text{ bytes} = 183 \text{ bytes} + 1 \text{ byte}$ (para torna-se múltiplo de 4). Logo, cada linha da matriz na memória deve ser capaz de armazenar 184 bytes (**unsigned char**)

No exemplo considerado anteriormente, o arquivo possui 56 linhas de pixels. Logo, a matriz criada em memória deve ter pelo menos 56 linhas.

PROCEDIMENTO DE COMPACTAÇÃO

Dividir a matriz “virtual” de pixels em 4 quadrantes. Para cada quadrante encontrado deve ser verificado se a quantidade de linhas de pixels ou colunas de pixels é inferior ou igual a 3 (base da recursão), caso isto ocorra então escolha o pixel central como representante desta região da imagem e armazene os dados RGB deste pixel em três vetores, vetor R, vetor G, vetor B.

Este procedimento de divisão da matriz de pixels em 4 quadrantes deve, OBRIGATORIAMENTE, ser recursivo.

Ao final deste processo os três vetores R, G, B contém as cores que representam cada parcela da imagem obtida pelo critério da base da recursão.

PROCEDIMENTO DE GERAÇÃO DO ARQUIVO COMPACTADO

Gravar em um arquivo de saída a sequência de bytes previamente armazenados nos seguintes vetores:

1º) vetor que armazena o cabeçalho BMP e cabeçalho Windows.

2º) vetores R, G, B.

Chame este arquivo de **imagemCompactada.zmp**

PROCEDIMENTO PARA LEITURA DO ARQUIVO COMPACTADO (imagemCompactada) e remontagem de um novo BMP.

1º) Leia os bytes iniciais do arquivo que armazena as informações referentes ao cabeçalho BMP armazene os bytes lidos no vetor de cabeçalho.

2º) Leia os bytes seguintes correspondentes aos dados dos vetores R, G, B.

3º) Realizar a divisão da matriz “virtual” de pixels em 4 quadrantes, assim como foi feito na etapa de compactação.

4º) No caso base da recursão deve ser realizado o preenchimento da região da matriz de bytes correspondente à imagem.

A primeira execução do caso base deve preencher a região da matriz de bytes com os elementos vetorR[0], vetorG[0], vetorB[0] para cada pixel desta região.

A segunda execução do caso base deve preencher a região da matriz de bytes com os elementos vetor[0], vetorG[1], vetorB[1] para cada pixel desta região.

E assim sucessivamente, até o término da função recursiva de remontagem da imagem.

5º) Ao final do preenchimento da matriz de bytes deve ser escrito o valor 0 (byte 00H) até o final de cada linha.

Por exemplo, em imagem em que cada linha da imagem possui 61 pixels deve-se escrever $61 \times 3 + 1 = 183 \text{ bytes} + 1 \text{ byte} = 184 \text{ bytes}$. (Deve-se completar com zero's no final de cada linha até que cada linha fique com uma quantidade de bytes que seja múltipla de 4. Neste exemplo, ao final da escrita de cada linha deve-se adicionar escreve um byte zero.

6º) Gravar o arquivo de remontagem da imagem chamado **imagemDescompactada.bmp**

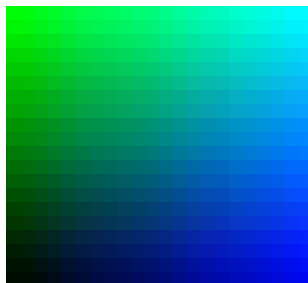
Ao final da execução do programa deve ser exibida na tela:

1. Quantidade de bytes do arquivo BMP original (**imagemOriginal.bmp**)
2. Quantidade de bytes do arquivo do arquivo compactado (**imagemCompactada.zmp**)
3. Quantidade de bytes do arquivo BMP reconstituído com perda de informação (**imagemDescompactada.bmp**)
4. Calcule a porcentagem de compactação referente à diferença de tamanho de armazenamento entre os arquivos **imagemOriginal.bmp** e **imagemCompactada.zmp**.
5. O nome e RA de cada integrante do grupo.

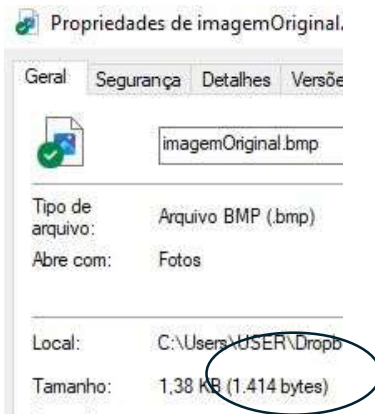
Após a execução do seu programa faça o teste de visualizar a imagem BMP gerada (**imagemDescompactada.bmp**)

UM EXEMPLO

Considere a seguinte **imagemOriginal.bmp** formada por 20 pixels x 22 pixels, padrão true color (24 bits para representar cada pixel). Representação ampliada abaixo para melhor visualização.



O arquivo **imagemOriginal.bmp** acima possui 1.414 bytes.



42	4D	86	05	00	00	00	00	00	00	36	00	00	00	28	00
00	00	16	00	00	00	14	00	00	00	01	00	18	00	00	00
00	00	50	05	00	00	13	0B	00	00	13	0B	00	00	00	00
00	00	00	00	00	00	00	00	00	0C	00	00	18	00	00	24
00	00	30	00	00	3C	00	00	48	00	00	55	00	00	61	00
00	6D	00	00	79	00	00	85	00	00	91	00	00	9D	00	00
AA	00	00	B6	00	00	C2	00	00	CE	00	00	DA	00	00	E6
00	00	F2	00	00	FF	00	00	00	00	0D	00	0C	0D	00	00
18	0D	00	24	0D	00	30	0D	00	3C	0D	00	48	0D	00	55
0D	00	61	0D	00	6D	0D	00	79	0D	00	85	0D	00	91	0D
00	9D	0D	00	AA	0D	00	B6	0D	00	C2	0D	00	CE	0D	00
DA	0D	00	E6	0D	00	F2	0D	00	FF	0D	00	00	00	00	1A
00	0C	1A	00	18	1A	00	24	1A	00	30	1A	00	3C	1A	00
48	1A	00	55	1A	00	61	1A	00	6D	1A	00	79	1A	00	85
1A	00	91	1A	00	9D	1A	00	AA	1A	00	B6	1A	00	C2	1A
00	CE	1A	00	DA	1A	00	E6	1A	00	F2	1A	00	FF	1A	00
00	00	00	28	00	0C	28	00	18	28	00	24	28	00	30	28
00	3C	28	00	48	28	00	55	28	00	61	28	00	6D	28	00

A matriz de pixels desta imagem possui 20 linhas X 22 colunas, conforme aponta o conteúdo do trecho inicial do arquivo.

Tamanho total do arquivo: (00 00 05 86)H = 1.414 bytes

Offset para o início da imagem: (00 36)H = 54 bytes

Linhas de pixels: (00 00 00 14)H = 20

Colunas de pixels: (00 00 00 16)H = 22

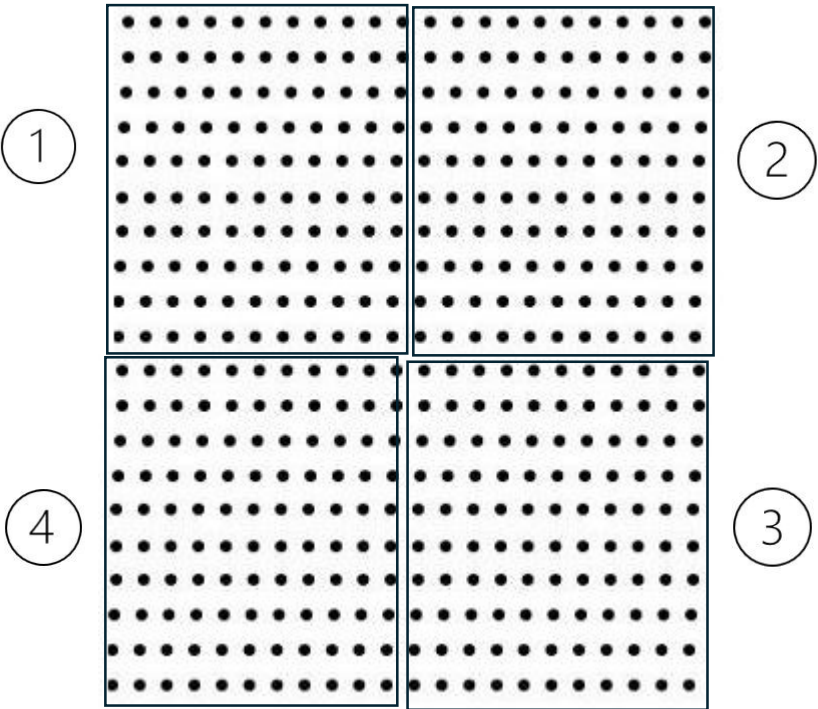
Está no formato 18H bits/pixel = 24 bits/pixel da imagem

Cada linha da imagem é apresentada por uma sequência de 22pixels*3bytes/pixel = 66 bytes, porém neste caso é necessário acrescentar 2 bytes e tem-se 66 + 2 = 68 bytes (68 é múltiplo de 4).

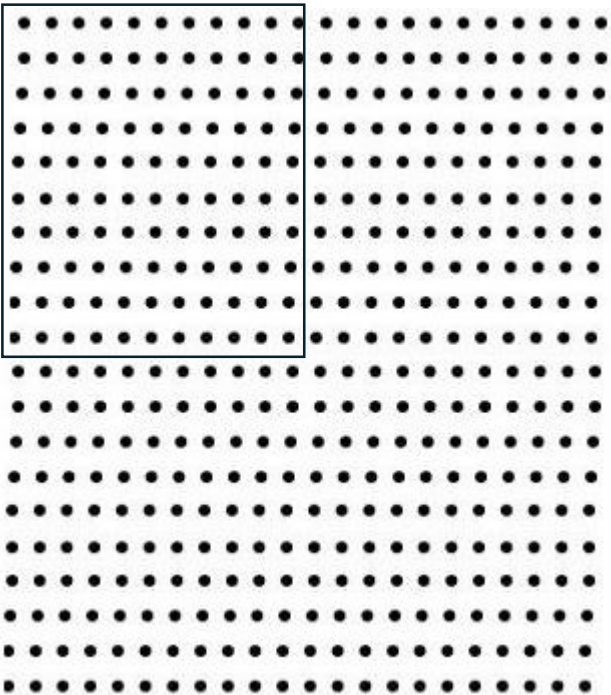
A representação dos pixels da imagem se inicia no byte 55. Logo após os cabeçalhos BMP (54 bytes).

Critério de parada adota neste exemplo é uma janela de 3 pixels ou menos na direção das linhas ou das colunas.

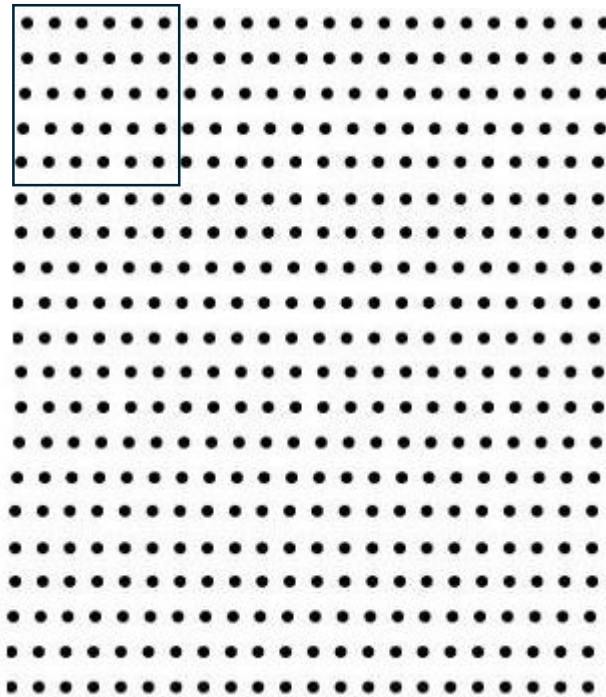
Abaixo a matriz esquemática de pixels da imagem (20 linhas x 22 colunas),



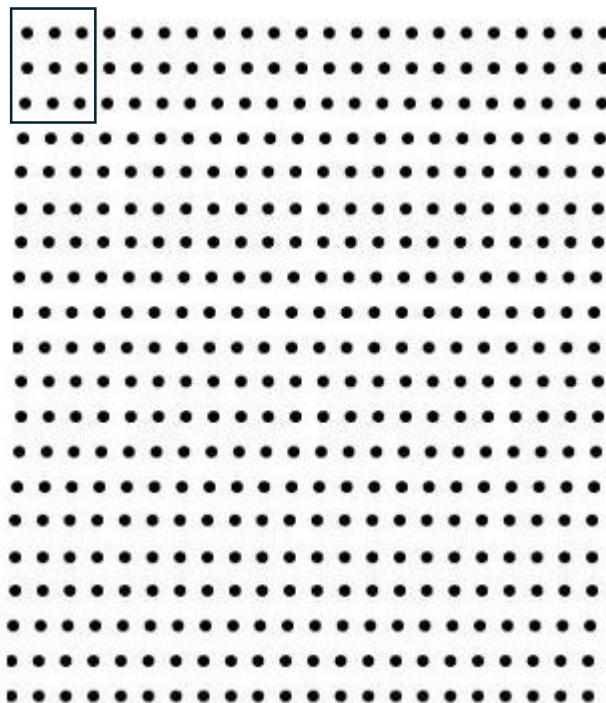
1ª chamada recursiva:



2ª chamada recursiva:



3ª chamada recursiva:



Atingida a base da recursão: Quantidade de pixels nas linhas ≤ 3 ou quantidade de pixels nas colunas ≤ 3 .

Vamos ver quais são os valores desses pixels desta parte da imagem:

	0	1	2
0	00 FF 00	0C FF 00	18 FF 00
1	00 F1 00	0C F1 00	F1 00 01
2	00 F4 00	0C E4 00	18 E4 00

Pixel representante desta base da recursão: **0C F1 00**

Insira nos vetores RGB os valores do pixel representante desta região da imagem:

vetorR[0] = 0C

vetorG[0] = F1

vetorB[0] = 00

E assim sucessivamente até o término das chamadas recursivas.

Grave o arquivo compactado (imagem.zmp) contendo a seguinte sequência de gravação:

Vetor do cabeçalho

vetorR índice 0

vetorG índice 0

vetorB índice 0

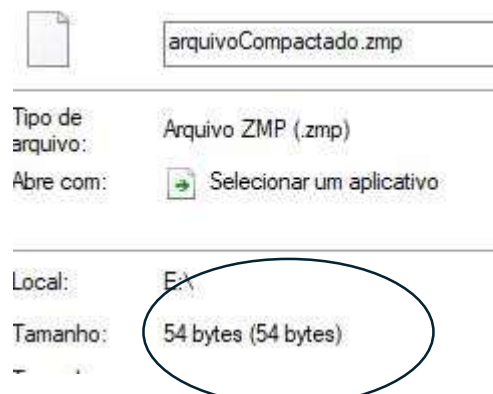
vetorR índice 1

vetorG índice 1

vetorB índice 1

Até o último índice dos vetores RGB que tenha sido utilizado para armazenar as cores dos pixels representantes da imagem.

O arquivo compactado gerado para esta imagem deste exemplo possui 54 bytes.



DESCOMPACTAÇÃO DA IMAGEM

Decompactar o **arquivoCompactado.zmp** conforme já detalhado e gerar o arquivo de saída **imagemDescompactada.bmp** para leitura da reconstrução da imagem Bitmap com perda de informação.

”4º) No caso base da recursão deve ser realizado o preenchimento da região da matriz de bytes correspondente à imagem. “

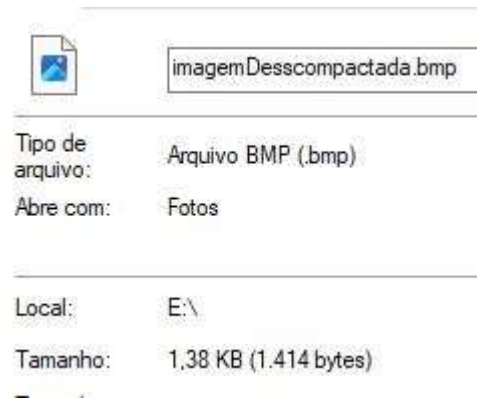
Neste exemplo, os elementos lidos do arquivo compactado referentes ao vetorR[0], vetorG[0], vetorB[0] serão escritos na região correspondente da matriz de bytes da imagem a ser reconstruída.

Escrever na matriz de bytes da imagem o pixel representante desta região:

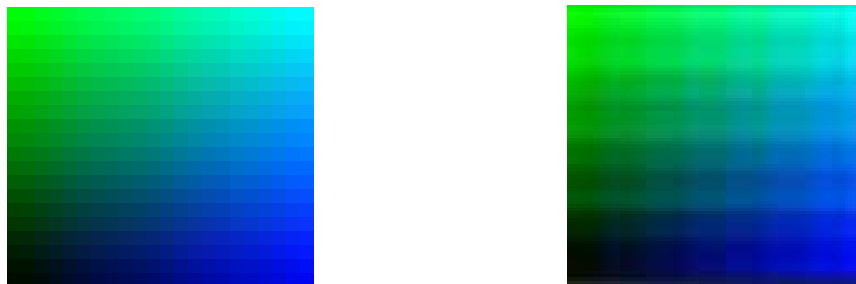
	0	1	2	3	4	5	6	7	8
0	0C	F1	00	0C	F1	00	0C	F1	00
1	0C	F1	00	0C	F1	00	0C	F1	00
2	0C	F1	00	0C	F1	00	0C	F1	00´

E assim, sucessivamente para os demais índices do vetor.

Note que o arquivo **imagemDescompactada.bmp** gerado a partir de **imagemCompactada.zmp** possui o mesmo tamanho da imagem original bmp.



Abaixo à esquerda a imagem original e à direita a imagem reconstruída a partir do arquivo compactado.



O QUE DEVE SER ENTREGUE

O código fonte do programa feito em linguagem C. O código fonte do programa deve ser postado no *moodle*.

Conjunto de 3 imagens BMP 24 true color. Cada conjunto contendo a imagem original (imagem.bmp), o arquivo compactado da imagem (imagemCompactada.zmp) e o arquivo remontado a partir da imagem (imagemDescompactada.bmp)

Um .DOCX com a impressão das imagens originais e reconstruídas.

O QUE SERÁ AVALIADO

- A corretude do programa (se o programa resolve o problema proposto)
- A originalidade da solução.

- Clareza do código que inclui identificação, comentários, nomes intuitivos para as variáveis.
- Nome e RA dos integrantes do grupo no cabeçalho do código fonte.
- Apresentação e execução do programa em laboratório conforme calendário das datas abaixo.

DATAS DE ENTREGA

Dia 25 de maio de 2025 no moodle.

Dia 26 de maio de 2025. Apresentação ao professor de laboratório, turma L12

Dia 28 de maio de 2025. Apresentação ao professor de laboratório, turma L11

OBSERVAÇÕES

A ausência da apresentação do projeto em laboratório, de acordo com o calendário acima, resulta em uma redução de 3,0 (três) pontos na nota final do projeto. Esta redução de nota é individual para cada integrante do grupo que não comparecer no dia da apresentação do trabalho no laboratório.