Atividade de Programação 1 (A1)

Atividade prévia 0.a - “Hello, C!”

Crie um arquivo chamado **ex0a_hello.c** e faça o programa imprimir “Hello, C!” usando:

#include <stdio.h>
printf(...) na função main()

Atividade prévia 0.b - Arrays e Alocação Dinâmica

Crie um arquivo chamado **ex0b_arrays.c** e declare duas formas de array:

Estático: *unsigned char arrEstatico[10];*
Dinâmico: *unsigned char *arrDinamico = malloc(10);*

Preencha ambos os arrays com valores de 0 a 9 usando ***for***. Imprima os valores dos dois arrays para verificar que estão preenchidos corretamente. Libere a memória do array dinâmico com *free(arrDinamico)*.

Atividade prévia 0.c - Indexação 2D e Função de Clamp

Este exercício simula, em pequeno escala, o que ocorre quando o código final percorre os pixels de uma imagem (cada pixel possui índices de linha e coluna).

Crie um arquivo chamado **ex0c_clamp2d.c**.

Implemente uma função auxiliar:

```
unsigned char clamp(float valor) {
    if (valor < 0.0f)   return 0;
    if (valor > 255.0f) return 255;
    return (unsigned char)valor;
}
```

Na função main(), defina duas variáveis para representar o “tamanho” da imagem, por exemplo:

```
int width = 4;
int height = 4;
```

(Opcional: você pode solicitar esses valores ao usuário via scanf.)

Aloque dinamicamente um array 1D para representar a imagem:
*unsigned char *img = malloc(width * height);*

Use dois laços for (um para row, outro para col).
Calcule o índice 1D como index = row * width + col;
Atribua a img[index] algum valor com base em (row + col) * 50.0f e use clamp(...) para converter em unsigned char.
Após preencher, imprima os valores em formato de matriz (4×4, por exemplo) para ver a distribuição.
Libere a memória com free(img).

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

unsigned char clamp(float valor) {
  if (valor < 0.0f) return 0;
  if (valor > 255.0f) return 255;
  return (unsigned char)valor;
}

int main() {
  int width = 4;
  int height = 4;

  // Aloca memória para "width * height" bytes
  unsigned char *img = malloc(width * height);
  if (!img) {
    printf("Erro: malloc falhou.\n");
    return 1;
  }

  // Preenche usando índices 2D
  for (int row = 0; row < height; row++) {
    for (int col = 0; col < width; col++) {
      int index = row * width + col;
      float valor = (row + col) * 50.0f;
      img[index] = clamp(valor);
    }
  }

  // Imprime em formato de matriz
  for (int row = 0; row < height; row++) {
    for (int col = 0; col < width; col++) {
      int index = row * width + col;
      printf("%3d ", img[index]);
    }
    printf("\n");
  }

  // Libera a memória
  free(img);

  return 0;
}
```

Atividade 1.a
Com base na Figura 1, implemente um programa na linguagem C utilizando o VS Code que leia um arquivo de imagem e aplique uma sequência de operações para extração das componentes (canais) de uma imagem colorida nos espaços de cores RGB (Red, Green, Blue) , CMY (Cyan, Magenta, Yellow) e HSV (Hue, Saturation, Value). O program deve salvar os canais extraídos em arquivos images diferentes. Use o arquivo de imagem colorida image1.png (Figura 1a) como entrada do programa e salve as imagens em escala de cinza das componentes em arquivos de extensão png. Nomeie estas imagens de saída do programa como image1_X.png, onde X é nome do canal (por exemplo, image1_red.png, image1_green.png, etc)

Restrições
É permitido **apenas** o uso das funções matemáticas e de leitura e escrita de arquivos de imagem disponíveis em bibliotecas de terceiros, as demais funções devem ser implementadas.

Forma de entrega
A entrega deve ser feita via Ensino Aberto. Entregar os arquivos com o código-fonte do programa e as imagens utilizadas e geradas pelo programa. Gere um arquivo compactado de extensão ZIP contendo todos os arquivos.

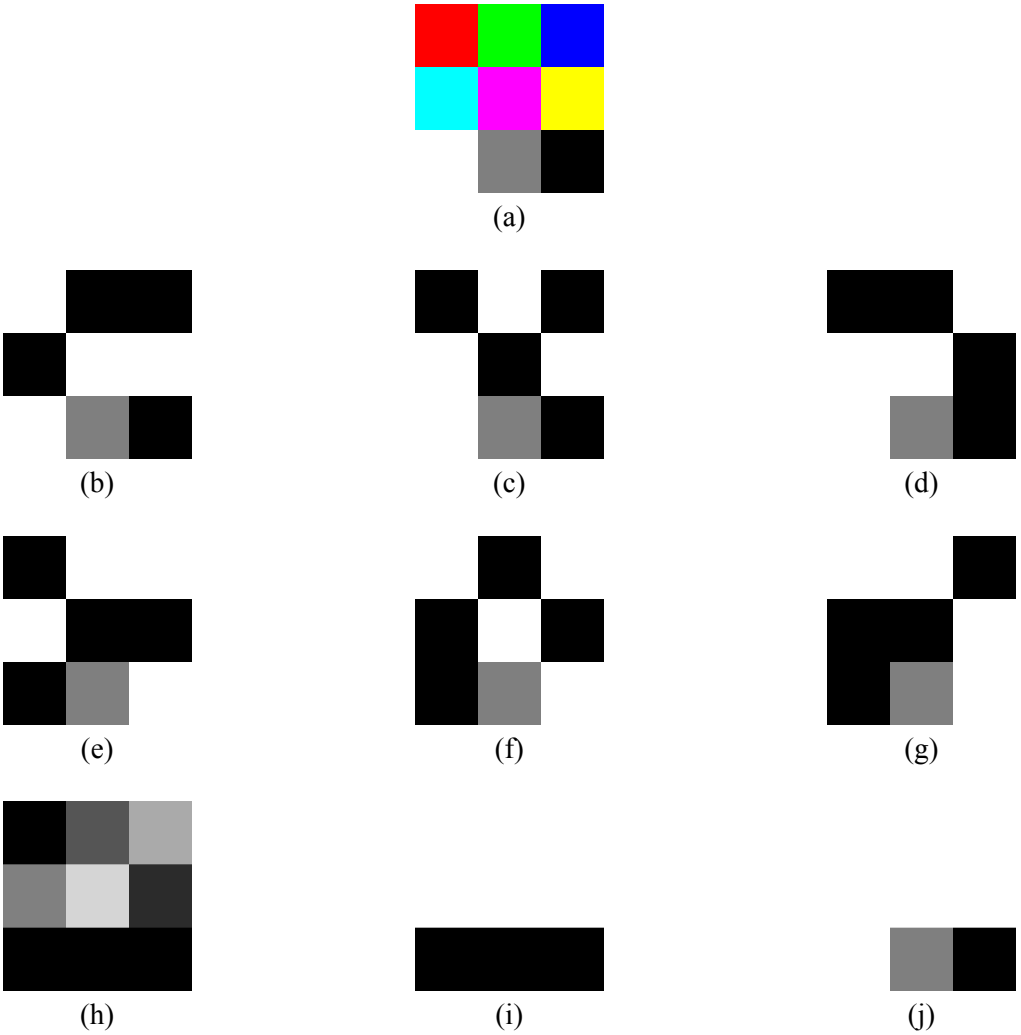

Figure 1: (a) image1.png; (b) image1_red.png; (c) image1_green.png; (d) image1_blue.png; (e) image1_cyan.png; (f) image1_magenta.png; (g) image1_yellow.png; (h) image1_hue.png; (i) image1_saturation.png; (j) image1_value.png